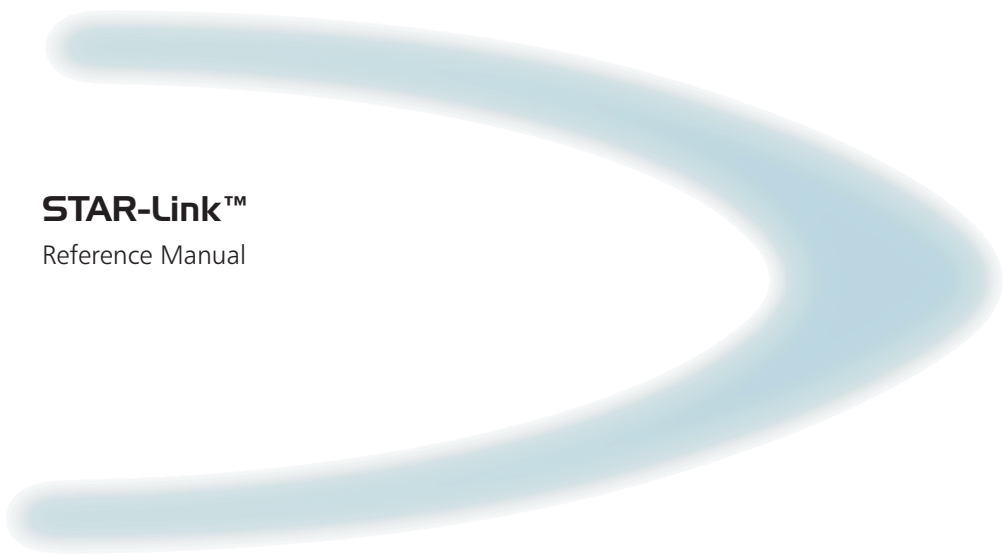




# STAR-Link™

Reference Manual





# **STAR-Link™**

## **REFERENCE MANUAL**





DATALOGIC S.p.A.  
Via Candini 2  
40012 - Lippo di Calderara di Reno  
Bologna - Italy

STAR-Link™

Ed.: 07/2003

This manual refers to software version 1.40 and later

**ALL RIGHTS RESERVED**

Datalogic reserves the right to make modifications and improvements without prior notification.

Datalogic shall not be liable for technical or editorial errors or omissions contained herein, nor for incidental or consequential damages resulting from the use of this material.

Product names mentioned herein are for identification purposes only and may be trademarks and or registered trademarks of their respective companies.

© Datalogic S.p.A. 2001 - 2003

17/07/03

## DATALOGIC S.p.A. Software License Agreement

This legal document is an agreement between you, the end user and DATALOGIC S.p.A. BY INSTALLING THE SOFTWARE, YOU ARE AGREEING TO BECOME BOUND BY THE TERMS OF THIS AGREEMENT, which includes the SOFTWARE LICENSE, LIMITED WARRANTY and ACKNOWLEDGMENT.

**GRANT OF LICENSE.** DATALOGIC grants to you the right to use one copy of the enclosed DATALOGIC S.p.A., program (the "SOFTWARE") on a single terminal connected to a single computer (i.e.; with a single CPU). You may not network the SOFTWARE or otherwise use it on more than one computer or computer terminal at the same time.

**COPYRIGHT.** The SOFTWARE is owned by DATALOGIC S.p.A. or its suppliers and is protected by copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material (i.e.; a book or musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the written materials accompanying the software.

**NON PERMITTED USES:** Without the express permission of DATALOGIC S.p.A., you may not:

1. Use the software in a computer service business including rental, networking or time sharing software, nor may you use it for multiple users, or multiple computer system applications in the absence of individual network licenses with DATALOGIC S.p.A.
2. Use, Copy or modify, alter or transfer, electronically or otherwise, the software or documentation except as expressly allowed in this agreement.
3. Translate, reverse engineer, de-assemble, de-compile or create derivative works based on the written materials.
4. Sub-license or lease this program or its documentation.

**LIMITED WARRANTY:** This software and accompanying written materials are provided "as is" without warranty of any kind. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THIS SOFTWARE IS ASSUMED BY YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE IN USE, YOU (AND NOT DATALOGIC OR ITS AGENTS) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION OF DATA. FURTHER, DATALOGIC DOES NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF, OR THE RESULTS OF THE USE OF THE SOFTWARE, IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE AND YOU RELY ON THE SOFTWARE AND RESULTS AT YOUR OWN RISK.

DATALOGIC warrants to the original licensee that the CD-Rom on which the SOFTWARE is recorded is free from defects in materials and workmanship under normal use and service for a period of ninety (90) days from the date of delivery as evidenced by a copy of your receipt. DATALOGIC's entire liability and your exclusive remedy shall be at DATALOGIC's option, replacement of the CD-Rom which shall be returned to DATALOGIC with a copy of your receipt. If failure of the CD-Rom has resulted from accident, abuse or misapplication of the product, then DATALOGIC shall have no responsibility to replace the product under this warranty.

THE ABOVE IS THE ONLY WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, THAT IS MADE BY DATALOGIC ON THIS DATALOGIC PRODUCT. IN NO EVENT SHALL DATALOGIC OR ITS SUPPLIERS, NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION OR DELIVERY OF THIS PRODUCT, BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE, THE RESULTS OF USE, OR INABILITY TO USE SUCH PRODUCT, EVEN IF DATALOGIC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**ACKNOWLEDGMENT:** You acknowledge that you have read this LICENSE and Limited Warranty, understand them and agree to be bound by their terms and conditions. You also agree that the LICENSE and Limited Warranty are the complete and exclusive statement between the parties and supersedes all other communications between the parties relating to the subject matter of the LICENSE or the Limited Warranty.

# CONTENTS

---

|          |                                              |           |
|----------|----------------------------------------------|-----------|
| <b>1</b> | <b>GENERAL INFORMATION .....</b>             | <b>1</b>  |
| 1.1      | Definition of the System .....               | 1         |
| 1.2      | Minimum Configuration Required .....         | 1         |
| 1.3      | STAR-Link™ Contents .....                    | 2         |
| 1.4      | Installing STAR-Link™ .....                  | 3         |
| 1.5      | Files Necessary to Use STAR-Link™ .....      | 3         |
| 1.6      | Conventions .....                            | 4         |
| <br>     |                                              |           |
| <b>2</b> | <b>USE AND FUNCTIONING .....</b>             | <b>5</b>  |
| 2.1      | Using STAR-Link™ ActiveX .....               | 5         |
| 2.2      | Broadcast and Factory-Loaded Addresses ..... | 6         |
| 2.3      | RF Device Address .....                      | 6         |
| 2.4      | Component Class Overview .....               | 8         |
| 2.4.1    | Main Control Object .....                    | 9         |
| 2.4.2    | Base Station Collections .....               | 11        |
| 2.4.3    | Device Collections .....                     | 11        |
| 2.4.4    | Base Station Object .....                    | 12        |
| 2.4.5    | Device Object .....                          | 13        |
| <br>     |                                              |           |
| <b>3</b> | <b>MAIN CONTROL OBJECT .....</b>             | <b>14</b> |
| 3.1      | Properties .....                             | 14        |
|          | Property <b>BaudRate</b> .....               | 14        |
|          | Property <b>CommPort</b> .....               | 15        |
|          | Property <b>Devices</b> .....                | 16        |
|          | Property <b>Mode</b> .....                   | 17        |
|          | Property <b>NetworkStatus</b> .....          | 20        |
|          | Property <b>Satellites</b> .....             | 21        |
|          | Property <b>Synchro</b> .....                | 22        |
|          | Property <b>Timeout</b> .....                | 23        |
|          | Property <b>Timer</b> .....                  | 24        |
|          | Property <b>Timer</b> .....                  | 24        |
| 3.2      | Methods .....                                | 25        |
|          | Method <b>CloseLine</b> .....                | 25        |
|          | Method <b>CloseNetwork</b> .....             | 26        |
|          | Method <b>GetControlVersion</b> .....        | 27        |
|          | Method <b>GetDLLVersion</b> .....            | 28        |
|          | Method <b>GetLostFrameBehaviour</b> .....    | 29        |
|          | Method <b>Load</b> .....                     | 30        |
|          | Method <b>LostFrame</b> .....                | 31        |
|          | Method <b>OpenLine</b> .....                 | 32        |
|          | Method <b>OpenNetwork</b> .....              | 33        |
|          | Method <b>Save</b> .....                     | 34        |
|          | Method <b>Save</b> .....                     | 34        |
|          | Method <b>SetLostFrameBehaviour</b> .....    | 35        |

|     |                                             |    |
|-----|---------------------------------------------|----|
| 3.3 | Events .....                                | 37 |
|     | Event <b>AliveEvent</b> .....               | 37 |
|     | Event <b>DataEvent</b> .....                | 38 |
|     | Event <b>DeviceInfoEvent</b> .....          | 40 |
|     | Event <b>ErrorEvent</b> .....               | 41 |
|     | Event <b>LostFrameEvent</b> .....           | 42 |
|     | Event <b>SatCfgEvent</b> .....              | 43 |
|     | Event <b>SleepSatEvent</b> .....            | 44 |
|     | Event <b>StatusEvent</b> .....              | 46 |
| 4   | <b>BASE STATION COLLECTION OBJECT</b> ..... | 47 |
| 4.1 | Properties .....                            | 47 |
|     | Property <b>Count</b> .....                 | 47 |
|     | Property <b>Item</b> .....                  | 48 |
|     | Property <b>Satellite</b> .....             | 49 |
| 4.2 | Methods .....                               | 50 |
|     | Method <b>Add</b> .....                     | 50 |
|     | Method <b>Clear</b> .....                   | 51 |
|     | Method <b>Remove</b> .....                  | 52 |
| 5   | <b>DEVICE COLLECTION OBJECT</b> .....       | 53 |
| 5.1 | Properties .....                            | 53 |
|     | Property <b>Count</b> .....                 | 53 |
|     | Property <b>DataLen</b> .....               | 54 |
|     | Property <b>Device</b> .....                | 56 |
|     | Property <b>DeviceData</b> .....            | 57 |
|     | Property <b>Item</b> .....                  | 59 |
| 5.2 | Methods .....                               | 60 |
|     | Method <b>Add</b> .....                     | 60 |
|     | Method <b>Clear</b> .....                   | 61 |
|     | Method <b>Remove</b> .....                  | 62 |
| 6   | <b>BASE STATION OBJECT</b> .....            | 63 |
| 6.1 | Properties .....                            | 63 |
|     | Property <b>BaudRateRF</b> .....            | 63 |
|     | Property <b>BaudRateRS</b> .....            | 64 |
|     | Property <b>BufferSize</b> .....            | 64 |
|     | Property <b>BufferThreshold</b> .....       | 64 |
|     | Property <b>Direct</b> .....                | 65 |
|     | Property <b>Enabled</b> .....               | 65 |
|     | Property <b>FrameLife</b> .....             | 66 |
|     | Property <b>Interface</b> .....             | 66 |
|     | Property <b>MaxBackOff</b> .....            | 67 |
|     | Property <b>MinBackOff</b> .....            | 67 |
|     | Property <b>PID</b> .....                   | 67 |
|     | Property <b>PowerSave</b> .....             | 68 |

|          |                                                        |     |
|----------|--------------------------------------------------------|-----|
|          | Property <b>PreambleLength</b> .....                   | 68  |
|          | Property <b>RfMaxProtocolVersion</b> .....             | 68  |
|          | Property <b>RfMinProtocolVersion</b> .....             | 70  |
|          | Property <b>RxFree, RxTotal, TxFree, TxTotal</b> ..... | 71  |
|          | Property <b>SatID</b> .....                            | 73  |
|          | Property <b>NewSatID</b> .....                         | 74  |
| 6.2      | Methods .....                                          | 75  |
|          | Method <b>Alive</b> .....                              | 75  |
|          | Method <b>GetConfiguration</b> .....                   | 76  |
|          | Method <b>GetStatus</b> .....                          | 78  |
|          | Method <b>SetConfiguration</b> .....                   | 80  |
| <b>7</b> | <b>DEVICE OBJECT</b> .....                             | 81  |
| 7.1      | Properties .....                                       | 81  |
|          | Property <b>BufferSize</b> .....                       | 81  |
|          | Property <b>DevID</b> .....                            | 81  |
|          | Property <b>Enabled</b> .....                          | 82  |
|          | Property <b>PID</b> .....                              | 83  |
|          | Property <b>RfMaxProtocolVersion</b> .....             | 84  |
|          | Property <b>RfMinProtocolVersion</b> .....             | 85  |
| 7.2      | Method .....                                           | 86  |
|          | Method <b>DataWrite</b> .....                          | 86  |
| <b>8</b> | <b>ERROR CODES</b> .....                               | 88  |
| 8.1      | Control Error Codes .....                              | 88  |
| 8.2      | Network Layer DLL Error Codes .....                    | 89  |
| 8.3      | Client Error Codes .....                               | 90  |
| 8.4      | STARGATE™ RF Base Station Error Codes .....            | 90  |
| <b>9</b> | <b>PROPERTY PAGES</b> .....                            | 91  |
| 9.1      | General Control Properties .....                       | 91  |
| 9.2      | Collection Properties .....                            | 92  |
| <b>A</b> | <b>PROGRAMMING EXAMPLES</b> .....                      | 93  |
| A.1      | STAR-Link™ Visual Basic 6.0 Example .....              | 93  |
| A.1.1    | Network Menu .....                                     | 95  |
| A.1.2    | Settings Menu .....                                    | 96  |
| A.1.3    | Action Menu .....                                      | 101 |
| A.1.4    | Help Menu .....                                        | 102 |
| A.1.5    | Warning Messages .....                                 | 103 |
| A.2      | STAR-Link™ Excel 97 Example .....                      | 104 |
| A.2.1    | Network Configuration .....                            | 105 |
| A.3      | STAR-Link™ Access 97 Example .....                     | 108 |
| A.3.1    | MainForm Mask .....                                    | 109 |
| A.3.2    | Other Masks .....                                      | 110 |
| A.4      | STAR-Link™ C++ Builder 5 Example .....                 | 111 |

A.5 STAR-Link™ Delphi 6 EXAMPLE ..... 112

B CONTROL INTERFACE..... 113



# 1 GENERAL INFORMATION

---

## 1.1 DEFINITION OF THE SYSTEM

Datalogic STAR-System™ provides a modern and flexible RF Narrow Band Solution for Mobile applications, exploiting one of the widest product range available on the market, including Hand-Held Readers and Portable Terminals.

STAR-System™ links all the RF devices and their own host applications providing a real time fast, reliable and cost-effective wireless connection between the operators and the most common Windows based software tools, the most familiar databases or the most widespread ERP/WMS systems.

Thanks to STAR-Link™ software tool, it is very easy to integrate data coming from the Datalogic STAR-System™ into an Excel spreadsheet or an Access database: even easier for a programmer is the development of a custom software to exchange data in real time with Formula RF terminals or Datalogic Hand-Held Readers using Rapid Application Development languages such as Visual Basic, C++Builder, Delphi, etc...

STAR-Link™ is a professional software tool addressed to programmers familiar both with "object programming" and 95/98/NT/ME/2000/XP environments. Its main component is an Automation COM object implementing a dual interface (it is possible to use it with Visual Basic and Office application) that manages, at a low level, the new Radio Frequency Network based on the new Datalogic CSMA/CA protocol.

## 1.2 MINIMUM CONFIGURATION REQUIRED

In order to install and use the STAR-Link™ packet, it is necessary to have at least the following:

- PC with a Pentium processor;
- one of the following operating systems:
  - Windows 95 - SR2 or later
  - Windows 98
  - Windows NT + Service Pack 4
  - Windows ME
  - Windows 2000
  - Windows XP
- free serial port;
- 16 MB memory for Windows 95/98, 32 MB for Windows NT/ME/2000/XP;
- hard disk with at least 25 MB free space to run;
- VGA video card or better.

### 1.3 STAR-Link™ CONTENTS

STAR-Link™ installation is available in the STAR-System Tools CD-ROM. It contains the following programs:

- STAR-Link™ ActiveX and Network Manager DLL;
- Examples based on STAR-Link™ ActiveX and developed in Excel 97, Access 97, Visual Basic 6.0, C++Builder 5.0 and Delphi 6.0 applications; their relative source files are provided, too; refer to the example developed in Visual Basic applications for a complete management of methods, properties and events;
- DL.EXE program to send firmware to Formula terminals when working in DOS environment;
- EasySend.EXE, a new program to send firmware to Formula terminals working in Windows environment;

and the following documents in .pdf format:

- STAR-Link™ Reference Manual;
- STAR★GATE™ Installation Manual;
- “Connect Gryphon™ M and Dragon™ M to STARGATE™” document;
- STAR-Box™ Installation Manual;
- STAR & Play™ Software Configuration Manual;
- STAR★Modem™ Reference Manual;
- “STAR★Modem™ Configuration Using Barcodes” booklet;
- “RF Device Setup for STAR★Modem™” document.

## 1.4 INSTALLING STAR-Link™

Once the CD-ROM is inserted, launch the Windows software STAR-Link™ x.xx.exe for WIN 95/98/NT/ME/2000/XP where "x.xx" corresponds to the software release. Then, click the Setup button from the opening window.

The following folders will be installed:

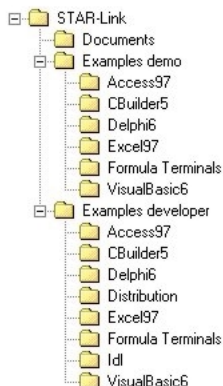


Figure 1 - Installation tree

Since the STAR-Link™ packet provides an ActiveX component that is registered in Operating System Registry, you must have administrator rights to successfully complete the installation in Windows NT, Windows 2000, or Windows XP.

## 1.5 FILES NECESSARY TO USE STAR-LINK™

Two files are necessary to use STAR-Link™:

- RFNCLI32.dll
- NRFCCTL.dll

where the first one is the network manager working directly with the serial port communication, while the second one is the ActiveX .dll.

NRFCCTL.dll is a COM component developed through ActiveX Template Library (ATL) using the Microsoft Visual C++ language.



### NOTE

*You must have administrator rights to use STAR-Link™ examples and to develop new applications in Windows XP environments.*

## 1.6 CONVENTIONS

The term "base station" indicates a base access point, while "device" indicates a generic reader or data collector of the radio network.

The terms "Satellite", "Satellite object", "Satellite collection" and "SatID" refer respectively to " Base station", "Base station software descriptor", "Base station collection" and "Base station address".

The term "DevID" refers to the device address.

The terms "STAR-Link™ ActiveX" and "STAR-Link™ control" refer to the NRFCCTL.dll file.

The term "network manager DLL" refers to the RFNCLI32.dll file.

The properties, methods and events of the five control objects are listed alphabetically.

**Bold text** indicates a term or keyword that should be used exactly as written. For example names of properties, methods and events should be written exactly as shown.

## 2 USE AND FUNCTIONING

---

### 2.1 USING STAR-Link™ ActiveX

The software component is an Automation COM object implementing a dual interface (it is possible to use it with Visual Basic and Office application).

It contains two kinds of object collection:

- Base stations collection
- Device collection

The collection of the base stations (**Base stations**) consists of single object base stations (**Base station**), while the collection of devices (**Devices**) is composed of single object devices. All the devices and base stations represent a group of objects through which it is possible to navigate by using particular instructions provided by the Visual Basic programming language. All objects in the control have a dual interface.

Five objects are provided by the control:

- Main control object
- Base station collections
- Device collections
- Base station object
- Device object

**NOTE**

*The Connection Point is another interface representing the events that the control sends to the container.*

For every object you can define properties, events and methods and therefore also for STAR-Link™:

- the properties of a control object are values defining its status. They are visible and can be changed in the "Properties Window" during the development of an application. Some properties are accessible while the program is running. It is usually possible to configure them although not always possible to read them;
- the events are indications transmitted by the STAR-Link™ controls to the application containing them;
- the methods are the operations that the object carry out.

## 2.2 BROADCAST AND FACTORY-LOADED ADDRESSES

The Datalogic base stations have a **factory-loaded address (2000) to be changed in the range 0-1999, since the network is only activated when all base station addresses are within this range of values.**

The broadcast address (**2047**) can be sent to a base station through the PC serial interface. This address is very useful to get the configuration and all information provided by the Alive, GetConfiguration, GetStatus and SetConfiguration Method from a base station having an unknown address.



**CAUTION**

*When requesting a base station configuration by using a broadcast address, only one base station must be connected within the network.*

## 2.3 RF DEVICE ADDRESS

For correct functioning, all the RF devices of STAR-System™ must be inserted in the Device Collection before starting the network data collection. Each RF device transmits its PID, RFMaxProtocolVersion and RFMinProtocolVersion correct values during its registration, since these properties are automatically set to 0 at network initialization.

The PID value of Datalogic RF products is listed in the following table:

| Product        | PID Value |
|----------------|-----------|
| Undefined      | 0         |
| Formula 734/RF | 1         |
| Formula 725/RF | 2         |
| Formula 660/RF | 3         |
| Gryphon™ M100  | 19761     |
| Gryphon™ M200  | 19762     |
| Dragon M101    | 20241     |
| STAR-Modem     | 20300     |

Once the network initialization has been completed, the RF device registration occurs during the first data transaction:

- HHR device (ex.: Gryphon™ M): the network initialization occurs each time a new configuration parameter is set;
- PDC device (ex.: F734-E/RF): the network initialization occurs using the corresponding library function (see the DS for Formula manual).

Once all the RF devices registration has been completed, it is strongly advised to save the current configuration (see par. A.1 for details). In this way, at the next application startup the RF device PID, RFMaxProtocolVersion, RFMinProtocolVersion properties will already be defined.

Otherwise, if not saved, at the next application startup all the RF devices will have to be registered again for correct system functioning.

**CAUTION**

*Even if an RF device is not manually inserted in the collection, it can be registered in the system and automatically added to the device collection. Since this process is notified by the DeviceInfoEvent, it is advised to define an automatic configuration saving in the DeviceInfoEvent. In this way, the configuration always provides the RF devices already registered in the system.*

## 2.4 COMPONENT CLASS OVERVIEW

The following is the UML class diagram of STAR-Link™ ActiveX where the boxes represent the classes and the lines indicate the relationship between them.

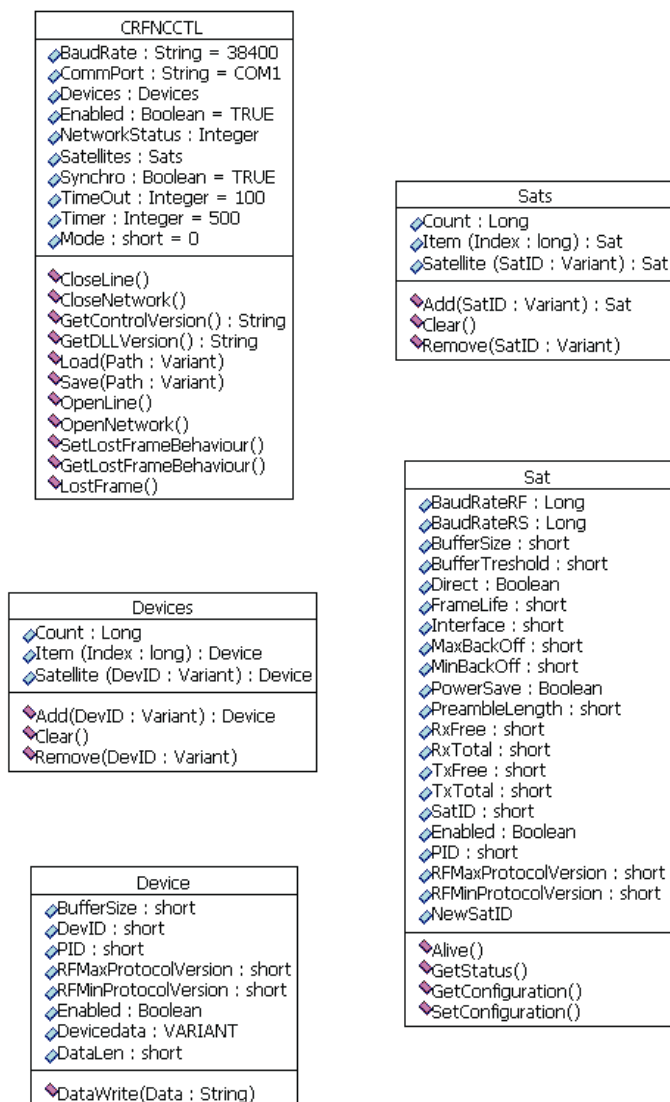


Figure 2 - Class overview

## 2.4.1 Main Control Object

### Control Properties

| <i>Property</i>      | <i>Description</i>                                                                      |
|----------------------|-----------------------------------------------------------------------------------------|
| <b>BaudRate</b>      | Sets transmission speed on serial line                                                  |
| <b>CommPort</b>      | Sets the communication port number                                                      |
| <b>Devices</b>       | User-definable collection of devices                                                    |
| <b>Mode</b>          | Defines the way to interpret data                                                       |
| <b>NetworkStatus</b> | Returns the web state, IDLE, RUNNING, STOP                                              |
| <b>Satellites</b>    | User-definable collection of base stations                                              |
| <b>Synchro</b>       | Sets the run-time mode of the control                                                   |
| <b>Timeout</b>       | Maximum waiting time after which it is assumed a frame has not arrived from serial line |
| <b>Timer</b>         | Sets the internal timer in asynchronous mode                                            |

### Control Methods

The control state, identified through its properties, is saved to the RFNCCTL.CFG configuration file. When defining the two collections, all the devices and the base stations (with their own state) are saved to the configuration file.

| <i>Method</i>                | <i>Description</i>                                                     |
|------------------------------|------------------------------------------------------------------------|
| <b>CloseLine</b>             | Closes the serial line                                                 |
| <b>CloseNetwork</b>          | Ends the network data collection on the serial line                    |
| <b>GetControlVersion</b>     | Returns the STAR-Link™ ActiveX version                                 |
| <b>GetDLLVersion</b>         | Returns the RFNCLI32.dll version                                       |
| <b>GetLostFrameBehaviour</b> | Gets the parameters to manage the lost frame behaviour                 |
| <b>Load</b>                  | Loads a precise control configuration from the <b>RFNCCTL.CFG</b> file |
| <b>LostFrame</b>             | Defines the lost frame management                                      |
| <b>OpenLine</b>              | Opens the serial line                                                  |
| <b>OpenNetwork</b>           | Starts the network data collection                                     |
| <b>Save</b>                  | Saves the present control configuration to a file                      |
| <b>SetLostFrameBehaviour</b> | Sets the parameters to manage the lost frame behaviour                 |

**Control Events**

| <b><i>Event</i></b>    | <b>Description</b>                                                                                                                                                                                                                                                                          |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>AliveEvent</b>      | Answers to the ALIVE command sent from a base station. It contains two pieces of information: the number of the base stations to which the Alive command was sent and a string of information about the base station                                                                        |
| <b>DataEvent</b>       | Sends to the container an event containing two pieces of information: the device identifier and the received data                                                                                                                                                                           |
| <b>DeviceInfoEvent</b> | Sent to the container when a device becomes part of the network                                                                                                                                                                                                                             |
| <b>ErrorEvent</b>      | Sends to the container an event containing two pieces of information: the error number and its description                                                                                                                                                                                  |
| <b>LostFrameEvent</b>  | Sent to the container when one or more devices loose a frame.                                                                                                                                                                                                                               |
| <b>SatCfgEvent</b>     | Sent to the container to reply to a configuration request of the base station. It contains the base station ID about which the information has been requested. Before starting the event, the control sets all the properties of that particular base station correctly.                    |
| <b>SleepSat</b>        | Sent to the container to communicate that a base station is in the Sleep state, that is it has not replied to the data collection request for several times.                                                                                                                                |
| <b>StatusEvent</b>     | It is an event sent to the container to reply to a status request of the base station. It contains the base station ID about which we asked the information. Before starting the event, the control sets the properties of about the number of bytes free for that particular base station. |

## 2.4.2 Base Station Collections

### Base Station Collection Properties

| <i>Property</i>  | <i>Description</i>                                                    |
|------------------|-----------------------------------------------------------------------|
| <b>Count</b>     | Gives the number of base station objects in the collection            |
| <b>Item</b>      | Retrieves a single base station object in the collection by the index |
| <b>Satellite</b> | Retrieves a single base station object in the collection by the ID    |

### Base Station Collection Methods

| <i>Method</i> | <i>Description</i>                                         |
|---------------|------------------------------------------------------------|
| <b>Add</b>    | Adds a base station object in the collection               |
| <b>Clear</b>  | Removes all base station objects from the collection       |
| <b>Remove</b> | Removes a specific base station object from the collection |

## 2.4.3 Device Collections

### Device Collection Properties

| <i>Property</i> | <i>Description</i>                                              |
|-----------------|-----------------------------------------------------------------|
| <b>Count</b>    | Gives the number of device objects in the collection            |
| <b>Device</b>   | Retrieves a single device object in the collection by the ID    |
| <b>Item</b>     | Retrieves a single device object in the collection by the index |

### Device Collection Methods

| <i>Method</i> | <i>Description</i>                                   |
|---------------|------------------------------------------------------|
| <b>Add</b>    | Adds a device object in the collection               |
| <b>Clear</b>  | Removes all the device objects from the collection   |
| <b>Remove</b> | Removes a specific device object from the collection |

## 2.4.4 Base Station Object

A base station object belongs to the collection of base stations and has defined properties typical of the radio protocol.

In details each base station sets out:

### Base Station Properties

| <i>Property</i>             | <i>Description</i>                                                |
|-----------------------------|-------------------------------------------------------------------|
| <b>BaudRateRF</b>           | Reserved                                                          |
| <b>BaudRateRS</b>           | Reserved                                                          |
| <b>BufferSize</b>           | Reserved                                                          |
| <b>BufferThreshold</b>      | Reserved                                                          |
| <b>Direct</b>               | Reserved                                                          |
| <b>Enabled</b>              | Indicates if a base station is enabled or disabled to be polled   |
| <b>FrameLife</b>            | Reserved                                                          |
| <b>Interface</b>            | Serial interface type (RS 232, RS 485)                            |
| <b>MaxBackOff</b>           | Reserved                                                          |
| <b>MinBackOff</b>           | Reserved                                                          |
| <b>PID</b>                  | Product ID (Datalogic convention to identify all the RF products) |
| <b>PowerSave</b>            | Reserved                                                          |
| <b>PreambleLength</b>       | Reserved                                                          |
| <b>RFMaxProtocolVersion</b> | Major radio protocol version                                      |
| <b>RFMinProtocolVersion</b> | Minor radio protocol version                                      |
| <b>RxFree</b>               | Free byte in the received buffer                                  |
| <b>RxTotal</b>              | Total bytes in the received buffer                                |
| <b>SatID</b>                | Base station ID (from 0 to 1999 included)                         |
| <b>TxFree</b>               | Free byte in the transmitted buffer                               |
| <b>TxTotal</b>              | Total bytes in the transmitted buffer                             |

### Base Station Methods

| <i>Method</i>           | <i>Description</i>                                                                        |
|-------------------------|-------------------------------------------------------------------------------------------|
| <b>Alive</b>            | Verifies whether the base station is connected or not                                     |
| <b>GetConfiguration</b> | Control to which the base station replies with its configuration                          |
| <b>GetStatus</b>        | Control to which the base station replies with its received and transmitted buffer state. |
| <b>SetConfiguration</b> | Sets the base station configuration starting from the base station state variables        |

### 2.4.5 Device Object

A device object belongs to the collection of devices. It also has some properties strictly depending on the radio protocol.

In details each device sets out:

#### Device Properties

| <b><i>Property</i></b>             | <b><i>Description</i></b>                                        |
|------------------------------------|------------------------------------------------------------------|
| <b><i>BufferSize</i></b>           | Device buffer length                                             |
| <b><i>DataLen</i></b>              | Gets data length in bytes                                        |
| <b><i>DeviceData</i></b>           | Gets data from a device                                          |
| <b><i>DevID</i></b>                | Device ID (from 0 to 1999 included)                              |
| <b><i>Enabled</i></b>              | Device enabled to send data to the application                   |
| <b><i>PID</i></b>                  | Product ID (Datalogic convention to identify all the RF product) |
| <b><i>RFMaxProtocolVersion</i></b> | Major radio protocol version                                     |
| <b><i>RFMinProtocolVersion</i></b> | Minor radio protocol version                                     |

#### Device Methods

| <b><i>Method</i></b>    | <b><i>Description</i></b>                        |
|-------------------------|--------------------------------------------------|
| <b><i>DataWrite</i></b> | Sends data to the device in the form of a string |

## 3 MAIN CONTROL OBJECT

---

An example written in Visual Basic is provided for all the properties and methods. In these examples we assume the control is named **RFNCCTL**.

### 3.1 PROPERTIES

#### *PROPERTY* **BAUDRATE**

##### *PURPOSE*

---

Communication port baud rate.

##### *DESCRIPTION*

---

This property sets or gets the transmission speed to be used in the network.

##### *IDL*

---

```
[propget, id(1), helpstring("property BaudRate")] HRESULT BaudRate([out, retval]
BSTR *pVal);
[propput, id(1), helpstring("property BaudRate")] HRESULT BaudRate([in] BSTR
newVal);
```

##### *DATA TYPE*

---

String

##### *POSSIBLE VALUES*

---

The control accept only these values:

9600, 19200, 38400, 57600, 115200

The default value is 38400

##### *EXAMPLE*

---

```
...
Dim BaudRate as String
...
RFNCCTL.BaudRate = "38400"
...
BaudRate = RFNCCTL.BaudRate
...
```

## *PROPERTY **COMMPORT***

### *PURPOSE*

---

Communication port selection.

### *DESCRIPTION*

---

This property sets or gets the communication port number of the PC.

### *IDL*

---

```
[propget, id(6), helpstring("property CommPort")] HRESULT CommPort([out, retval]  
BSTR *pVal);  
[propput, id(6), helpstring("property CommPort")] HRESULT CommPort([in] BSTR  
newVal);
```

### *DATA TYPE*

---

String

### *POSSIBLE VALUES*

---

The control accepts only these values:  
from COM1 to COM32  
The default value is COM1

### *EXAMPLE*

---

```
...  
Dim CommPort as String  
...  
RFNCCTL.CommPort = "COM1"  
...  
CommPort = RFNCCTL.CommPort  
...
```

## PROPERTY **DEVICES**

### PURPOSE

---

Gets a device collection object.

### DESCRIPTION

---

This property allows the developer to manage the device collection. It returns the object devices.

### IDL

---

```
[propget, id(3), helpstring("property Devices")] HRESULT Devices([out, retval]  
Idevices* *pVal);
```

### DATA TYPE

---

Object devices

### POSSIBLE VALUES

---

Read only property

### EXAMPLE

---

```
...  
Dim DevicesObject as Devices  
...  
Set DevicesObject = RFNCCTL.Devices  
  
' Add device to the collection  
DevicesObject.Add(1000)  
  
' Remove Devices from the collection  
DevicesObject.Remove(1500)  
...
```

## PROPERTY **MODE**

### PURPOSE

---

Defines the way to interpret data received from a device. If the transaction is in 2way mode (see DataEvent event) it also defines the way to write data to be sent to a device.

### DESCRIPTION

---

This property allows choosing between the following data interpretations in DataEvent:

- Text mode (default);
- Binary mode;
- Text mode with OEM-ANSI character conversion for data to and from device;
- Text mode with OEM-ANSI character conversion for data from device to application only;
- Text mode with OEM-ANSI character conversion for data to device only.

**The three text modes performing OEM-ANSI character conversion are available starting from STAR-Link™ rel. 1.21.**

The **ANSI** set of characters is essentially equivalent to the Windows set used in Windows programming.

The **OEM** set of characters is typically used in full-screen MS-DOS sessions for screen display. Characters in the decimal range 32 - 127 in Windows environment are usually the same as in OEM. The other characters in OEM set of characters (in the decimal range 0 to 31 and 128 to 255) correspond to the characters that can be displayed in a full-screen MS-DOS session. These characters are generally different from the Windows characters.

In all text modes, with or without OEM-ANSI conversion, when the control notifies the container of the data event, it sets data in Device.DeviceData (a VARIANT type) Bstring format (VT\_BSTR). The string is NULL terminated.

In the binary mode, the Device.DeviceData is an array of bytes, of Device.DataLen length (VT\_ARRAY | VT\_UI1).

Binary mode has been introduced to allow transmission of particular characters, such as NULL character.

### Example

A. Displaying a text containing the word "Françoise"

If sending data by setting the PropertyMode to "text mode", it may occur that a character different from "ç" will be displayed on the device screen. This is due to the fact that the device uses an OEM set of characters, while the Windows program uses the ANSI one. By setting the PropertyMode to "text mode with OEM-ANSI character conversion for data to device only" it is possible to display the desired character.

B. Receiving and displaying a text containing the word "Español"

If receiving data by setting the PropertyMode to "text mode", it may occur that a character different from "ñ" will be displayed on the PC video. This is due to the fact that the device uses an OEM set of characters, while the Windows program uses the ANSI one. By setting the PropertyMode to "text mode with OEM-ANSI character conversion for data from device to application only" it is possible to display the desired character.

By using one of the application programs provided with STAR-Link™ (ex. Visual Basic), it is possible to test the system behaviour.

### IDL

```
[propget, id(21), helpstring("property Mode")] HRESULT Mode([out, retval] short *pVal);  
[propput, id(21), helpstring("property Mode")] HRESULT Mode([in] Short newVal);
```

### DATA TYPE

Short

---

*POSSIBLE VALUES*

---

The control accepts only these values:

0 for text mode,

1 for binary mode,

2 for text mode with OEM-ANSI character conversion for data to and from device,

3 for text mode with OEM-ANSI character conversion for data from device to application only,

4 for text mode with OEM-ANSI character conversion for data to device only.

The default value is 0

---

*EXAMPLE*

---

```
...  
Public Const MODE_TEXT = 0  
Public Const MODE_BINARY = 1  
Dim mode As Integer  
...  
RFNCCTL.Mode = MODE_BINARY  
...  
mode = RFNCCTL.Mode  
...
```

## PROPERTY **NETWORKSTATUS**

### PURPOSE

---

Returns the network status.

### DESCRIPTION

---

This read only property returns the network status.

### IDL

---

[propget, id(13), helpstring("property NetworkStatus")] HRESULT NetworkStatus([out, retval] short \*pVal);

### DATA TYPE

---

Short

### POSSIBLE VALUES

---

The control return only these values:

NRF\_IDLE = 0

NRF\_LINE\_OPEN = 1

NRF\_NET\_OPEN = 2

NRF\_NET\_CLOSE = 3

### EXAMPLE

---

```
...
Public Const NRF_IDLE = 0           ' Network RF Status
Public Const NRF_LINE_OPEN = 1
Public Const NRF_NET_OPEN = 2
Public Const NRF_NET_CLOSE = 3
```

```
...
Dim strStatus As String
Select Case RFNCCTL.NetworkStatus
Case NRF_IDLE
    strStatus = "Network is in IDLE state"
Case NRF_LINE_OPEN
    strStatus = "Serial Line is OPENED"
Case NRF_NET_OPEN
    strStatus = "Network is RUNNING"
Case NRF_NET_CLOSE
    strStatus = "Network is in CLOSED"
End Select
MsgBox strStatus, vbExclamation, "Network Status"
...
```

## *PROPERTY **SATELLITES***

### *PURPOSE*

---

Gets a base station collection object.

### *DESCRIPTION*

---

This property allows the developer to manage the base station collection. It returns the object base stations.

### *IDL*

---

```
[propget, id(2), helpstring("property Satellites")] HRESULT Satellites([out, retval]  
ISats* *pVal);
```

### *DATA TYPE*

---

Object base stations

### *POSSIBLE VALUES*

---

Read only property

### *EXAMPLE*

---

```
...  
Dim SatellitesObject as Satellites  
...  
Set SatellitesObject = RFNCCTL.Satellites  
  
' Add base station to the collection  
SatellitesObject.Add(1000)  
  
' Remove base station from the collection  
SatellitesObject.Remove(1500)  
...
```

## *PROPERTY **SYNCHRO***

### *PURPOSE*

---

Way to send data to the container.

### *DESCRIPTION*

---

This property allows choosing between synchronized or asynchronous data events. In the first case the control notifies the container the data event, and waits until the application has proceeded the message. In the second case, the ActiveX sends the event and goes on.

The first mode is implemented by `SendMessage(...)`, the second mode by `PostMessage( ... )`.

### *IDL*

---

```
[propget, id(15), helpstring("property Synchro")] HRESULT Synchro([out, retval]  
    BOOL *pVal);  
[propput, id(15), helpstring("property Synchro")] HRESULT Synchro([in] BOOL  
    newVal);
```

### *DATA TYPE*

---

Bool

### *POSSIBLE VALUES*

---

The control accepts only these values:

TRUE, FALSE

The default value is TRUE.

### *EXAMPLE*

---

```
...  
Dim Synchro as Boolean  
...  
RFNCCTL.Synchro = TRUE  
...  
Synchro = RFNCCTL.Synchro  
...
```

## *PROPERTY **TIMEOUT***

### *PURPOSE*

---

Communication port timeout on received frames.

### *DESCRIPTION*

---

This property sets or gets the communication port timeout, which is the maximum waiting time after that we assume the frame does not arrive.

### *IDL*

---

```
[propget, id(7), helpstring("property Timeout")] HRESULT Timeout([out, retval] short *pVal);  
[propput, id(7), helpstring("property Timeout")] HRESULT Timeout([in] short newVal);
```

### *DATA TYPE*

---

Short

### *POSSIBLE VALUES*

---

The control accepts only these values (in millisecond):

10, 20, 50, 100, 200, 500, 1000, 2000

The default value is 100 ms. For a better output it is advised to set timeout values >100 ms.

### *EXAMPLE*

---

```
...  
Dim Timeout as integer  
...  
RFCCTL.Timeout = 200  
...  
Timeout = RFCCTL.Timeout  
...
```

## *PROPERTY **TIMER***

### *PURPOSE*

---

Time between two consecutive data events.

### *DESCRIPTION*

---

When you select the Asynchro operating mode, this property allows selecting the minimum time between two consecutive data events sent from the ActiveX to the container.

### *IDL*

---

```
[propget, id(16), helpstring("property Timer")] HRESULT Timer([out, retval] short *pVal);  
[propput, id(16), helpstring("property Timer")] HRESULT Timer([in] short newVal);
```

### *DATA TYPE*

---

Short

### *POSSIBLE VALUES*

---

The control accept only the values in 100, 2000 range:  
The default value is 500.

### *EXAMPLE*

---

```
...  
Dim iTimer as Integer  
...  
RFNCCTL.Timer = 100  
...  
iTimer = RFNCCTL.Timer  
...
```

## 3.2 METHODS

### *METHOD **CLOSELINE***

#### *PURPOSE*

---

Closes the serial line.

#### *DESCRIPTION*

---

Closes the serial line. This is the last operation the developer must do before terminating the application. For this reason, when the CloseLine method is preceded by the CloseNetwork one, it is advised to verify that the data collection has been completed correctly. Proceed by checking the NetworkStatus property value. If it is set to the NRF\_NET\_CLOSE=3 value (data collection completed), then, it is possible to call the CloseLine method. (See the source code in the examples provided).

#### *IDL*

---

```
[id(9), helpstring("method CloseLine")] HRESULT CloseLine();
```

#### *INPUT PARAMETERS*

---

None

#### *OUTPUT VALUE*

---

None

#### *EXAMPLE*

---

```
...  
' Set the serial line and open it  
RFNCCTL.CommPort = "COM2"  
RFNCCTL.BaudRate = "38400"  
RFNCCTL.Timeout = 100  
RFNCCTL.OpenLine  
...  
RFNCCTL.CloseLine  
...
```

## *METHOD* **CLOSENETWORK**

### *PURPOSE*

---

Stops the data collection.

### *DESCRIPTION*

---

This method allows stopping the data collection from base stations. It is necessary to close the network before terminating the application. This method closes the serial line too, and frees all the opened handles.

### *IDL*

---

```
[id(5), helpstring("method CloseNetwork")] HRESULT CloseNetwork();
```

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

None

### *EXAMPLE*

---

```
...  
Private Sub Form_Unload(Cancel As Integer)  
' Close the network when Unload Event is Invoked  
    ...  
    If RFNCCTL.NetworkStatus <> NRF_NET_OPEN Then  
        RFNCCTL.CloseNetwork  
    End If  
    ...  
End Sub
```

---

**METHOD *GetControlVersion***

---

**PURPOSE**

Retrieves the NRFCCTL.DLL version.

---

**DESCRIPTION**

Retrieves the string indicating the NRFCCTL.DLL version.

---

**IDL**

---

```
[id(17),helpstring("method  GetControlVersion")] HRESULT  GetControlVersion  
([out,retval] BSTR *Version);
```

---

**INPUT PARAMETERS**

None

---

**OUTPUT VALUE**

Version as a String

---

**EXAMPLE**

---

```
...  
' Show Control Version  
MsgBox RFNCCTL.GetControlVersion, vbExclamation, "CTL Version"  
...
```

## *METHOD **GetDLLVersion***

### *PURPOSE*

---

Retrieves the RFNCLI32.DLL version.

### *DESCRIPTION*

---

Retrieves the string indicating the RFNCLI32.DLL version.

### *IDL*

---

```
[id(14),helpstring("method GetDLLVersion")] HRESULT GetDLLVersion([out,retval]  
BSTR *Version);
```

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

The version as a String

### *EXAMPLE*

---

```
... Show DLL Version  
MsgBox RFNCCTL.GetDLLVersion, vbExclamation, "DLL  
Version"  
...
```

## METHOD **GetLostFrameBehaviour**

### PURPOSE

---

Gets the parameters of the Lost Frame Behaviour

### DESCRIPTION

---

See **SetLostFrameBehaviour** on page 34 for details.

### IDL

---

```
[id(19),      helpstring("method      GetLostFrameBehaviour")]      HRESULT  
GetLostFrameBehaviour([out]VARIANT_BOOL *Send2PC, [out]short* TimeNotify,  
[out]short* TimeGoingOn);
```

### INPUT PARAMETERS

---

None

### OUTPUT VALUE

---

Send2PC as Boolean  
TimeNotify as short  
TimeGoingOn as short

### EXAMPLE

---

```
...  
Dim SendToPC as Boolean  
Dim TimeNotify as Integer  
Dim TimeGoingOn as Integer  
' Get Lost Frame Behaviour  
RFNCCTL.GetLostFrameBehaviour SendToPC, TimeNotify, TimeGoingOn  
...
```

## *METHOD **LOAD***

### *PURPOSE*

---

Loads the control configuration parameters.

### *DESCRIPTION*

---

This method allows loading the control configuration parameters from the Hard Disk. This method (with the **Save** method) implements a kind of persistence of the control. All the control properties and both the Device and Base station collection are stored on the Hard Disk.

### *IDL*

---

[id(11), helpstring("method Load")] HRESULT Load([optional]VARIANT Path);

### *INPUT PARAMETERS*

---

VARIANT Path

It is the path of the directory where the control can find and read its configuration file (**RFNCCTL.CFG**).

### *OUTPUT VALUE*

---

None

### *EXAMPLE*

---

```
...  
' Load control parameters and open line  
RFNCCTL.Load "C:\ RFNCCTL\Settings"  
RFNCCTL.OpenLine  
RFNCCTL.OpenNetwork  
...
```

## *METHOD **LostFrame***

### *PURPOSE*

---

Defines the lost frame management.

### *DESCRIPTION*

---

Called when a **LostFrameEvent** has occurred, it defines the way to manage the lost frame. See **SetLostFrameBehaviour** on page 34 for details.

### *IDL*

---

```
[id(20), helpstring("method LostFrame")] HRESULT LostFrame([in]VARIANT_BOOL  
Send2PC, [out,retval]VARIANT_BOOL* retVal);
```

### *INPUT PARAMETERS*

---

Send2PC as BOOL

### *OUTPUT VALUE*

---

RetVal as BOOL

### *EXAMPLE*

---

```
...  
' Call the Lost Frame Method  
if Not RFNCCTL.LostFrame(True) = True Then  
    MsgBox "Error Calling Lost Frame ..", vbCritical, "Error"  
End If  
...
```

## *METHOD **OPENLINE***

### *PURPOSE*

---

Opens the serial line.

### *DESCRIPTION*

---

Opens the serial line. This is the first operation the developer must do. You can send all the base station commands, like **Alive**, **GetStatus ecc.** after opening the serial line. It is not necessary the network data collection is started.

### *IDL*

---

```
[id(8), helpstring("method OpenLine")] HRESULT OpenLine();
```

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

None

### *EXAMPLE*

---

```
...  
' Set the serial line and open it  
RFNCCTL.CommPort = "COM2"  
RFNCCTL.BaudRate = "38400"  
RFNCCTL.Timeout = 100  
RFNCCTL.OpenLine  
...
```

## *METHOD* **OPENNETWORK**

### *PURPOSE*

---

Starts the data collection.

### *DESCRIPTION*

---

This method allows starting the data collection from base stations. It is necessary to open the serial line before calling this method, otherwise the control returns an error. After opening the network the collected data is sent to the application.

### *IDL*

---

[id(4), helpstring("method OpenNetwork")] HRESULT OpenNetwork();

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

None

### *EXAMPLE*

---

```
...  
' Set the serial line and open it  
RFNCCTL.CommPort = "COM2"  
RFNCCTL.BaudRate = "38400"  
RFNCCTL.Timeout = 100  
RFNCCTL.OpenLine  
RFNCCTL.OpenNetwork  
...
```

## METHOD **SAVE**

### PURPOSE

---

Saves the control parameters to the Hard Disk.

### DESCRIPTION

---

This method allows saving to the Hard Disk the control configuration parameters. This method (with the **Load** method) implements a kind of persistence of the control. You can save more than one configuration file, and then load the correct file you need.

### IDL

---

```
[id(12), helpstring("method Save")] HRESULT Save([optional]VARIANT Path);
```

### INPUT PARAMETERS

---

VARIANT Path

It is the path of the directory where the control can store its configuration file (**RFNCCTL.CFG**).

### OUTPUT VALUE

---

None

### EXAMPLE

---

```
...  
' Setting control parameters  
RFNCCTL.CommPort = "COM1"  
RFNCCTL.BaudRate = "38400"  
RFNCCTL.Timeout = 100  
RFNCCTL.Synchro = FALSE  
RFNCCTL.Timer = 200  
  
' Save control parameters  
RFNCCTL.Save "C:\ RFNCCTL\Settings"  
...
```

## METHOD **SetLostFrameBehaviour**

### PURPOSE

---

Sets the parameters managing the lost frames.

Even if data frames are numbered, they might be sent to the host PC in a wrong order due to roaming (see chapter 4 in STARGATE™ installation manual for more details about the RF system). One of the main tasks of STAR-Link™ is to sort frames so that they are correctly sent to the application program.

### Example

STAR-Link™ waits for the frame number 5 from the RF device with address 1. If the frame number 6 is received before the frame number 5, the former is buffered and not sent to the application program. Once the frame number 5 is received, it is sent in the correct order to the application program together with frame number 6.

In case the base reset or the power going down accidentally causes the frame number 5 loss, STAR-Link™ keeps on waiting for the lost frame while buffering the other frames and stopping to send data from the device 1 to the application.

Thanks to the **TimeNotify** property it is possible to define a time after which the network manager DLL starts a procedure making sure that frame 5 is not present in any base station considering it as a lost frame.

When the missing frame is not found, the **LostFrameEvent** is raised. Then, using the **LostFrame** method it is possible to define if sending the buffered frames to the application or deleting them.

When the RF system is unattended and there is no user defining its behaviour in case of lost frames, two parameters are provided to program the system.

The **SendToPC** parameter allows sending buffered frames to the PC or deleting them; while the **TimeGoingOn** parameter allows setting a timeout after which the system behaviour defined by the **SendToPC** parameter is automatically forced.

### DESCRIPTION

---

**SendtoPC:** it is the default behaviour when one or more frame losses have occurred. If it is set to TRUE and it has not received an answer within the TimeGoingOn time, the base station frames following the lost one are sent to the container. On the other hand, by setting the method to FALSE, the frames are deleted.

**TimeNotify:** it is the time (given in *minutes*) after which STAR-Link™ starts a procedure making sure that one or more frames are lost. It starts counting from the last frame sent from the RF device to the container. Default and suggested value is 1. Possible values: bigger than 0.

**TimeGoingOn:** it is the timeout (given in *minutes*) set to wait for the user's answer before applying the *sendtoPC* defined behaviour. If TimeGoingOn is 0, STAR-Link™ keeps on waiting. If there has been a frame loss, the host network data managing will hang on.

---

*IDL*

---

```
[id(18), helpstring("method SetLostFrameBehaviour")] HRESULT  
SetLostFrameBehaviour(VARIANT_BOOL Send2PC, short TimeNotify, short  
TimeGoingOn);
```

---

*INPUT PARAMETERS*

---

Send2PC as Boolean  
TimeNotify as short  
TimeGoingOn as short

Default parameters are:

***SendtoPC*** = TRUE

***TimeNotify*** = 1

***TimeGoingOn*** = 10

---

*OUTPUT VALUE*

---

None

---

*EXAMPLE*

---

```
...  
' Set Lost Frame Behaviour  
RFNCTL.SetLostFrameBehaviour True, 0, 10  
...
```

### 3.3 EVENTS

#### *EVENT* **ALIVEEVENT**

---

##### *PURPOSE*

Alive answer.

---

##### *DESCRIPTION*

This event notifies the container when a base station answers to an alive command sent from the application. The base station answers with a string having the following format:

"**DL STARGATE x.y**" where x.y is the base station application version.

---

##### *IDL*

[id(3), helpstring("method AliveEvent")] HRESULT AliveEvent(short SatID, BSTR Info);

---

##### *INPUT PARAMETERS*

None

---

##### *OUTPUT VALUE*

SatID as short with the ID of the base station sending the alive answer  
Info as String containing information about the base station.

---

##### *EXAMPLE*

```
...  
Private Sub OCX_AliveEvent(ByVal SatID As Integer, ByVal Info  
As String)  
    MsgBox Info, vbExclamation, "Base station:" & CStr(SatID)  
End Sub...
```

## *EVENT DATAEVENT*

### *PURPOSE*

---

Data received

### *DESCRIPTION*

---

This event notifies the container when data has arrived. Data is available in DeviceData and DataLen properties of the device object identified by the address defined in DevID (see also property Mode of the Main Control object);

### *IDL*

---

[id(2), helpstring("method DataEvent")] HRESULT DataEvent(short SatID, short DevID, BOOL Kind);

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

SatID as short with the ID of the base station collecting data

DevID as short with the ID of the device sending data

Kind as Boolean: if TRUE indicates that the frame is two ways and that the device is waiting for an answer from the PC.

If FALSE the frame is one way.

**EXAMPLE**

```

...
Public Const MODE_TEXT = 0
Public Const MODE_BINARY = 1
...
Private Sub RFNCCTL_DataEvent(ByVal SatID As Integer, ByVal
DevID As Integer, ByVal Kind As Boolean)
On Error Resume Next
Dim item As ListItem
Dim sDevice As Device
Dim strAnswer As String
Dim vaData As Variant
Dim strData As String
Dim byteData As Byte
Dim i As Integer

    Set sDevice = OCX.Devices.Devices(DevID)
    vaData = sDevice.DeviceData
    ,
    'Data interpretation
    ,
    strData = ""
    If OCX.Mode <> MODE_BINARY Then
        strData = CStr(vaData)
    Else
        For i = 0 To sDevice.DataLen - 1
            byteData = vaData(i)
            strData = strData & Right("000" + CStr(byteData),3) + ""
        Next i
    End if

    Set item = ListData.ListItems.Add(, , CStr(nItem))
    item.ListSubItems.Add , , CStr(DevID)
    item.ListSubItems.Add , , strData
    ,
    'Answer manager
    ,
    If CBool(Kind) = True Then
        item.ListSubItems.Add , , "Two Way"
        If Not TheSameAnswer Then
            strAnswer = AnswerData
        Else
            'Send back the same data packet
            strAnswer = strData
        End If
        If Not sDevice.DataWrite(strAnswer, Len(strAnswer)) = True Then ListData.ListItems.Add
        , , "Error Send Data"
    Else
        item.ListSubItems.Add , , "One Way"
        End If
    End Sub

```

## *EVENT DEVICEINFOEVENT*

### *PURPOSE*

---

Device coming into the network.

### *DESCRIPTION*

---

This event will be sent from the control to the application when a new device becomes active in the network. The device must link to the network before sending data. In this way it is possible to capture the particular frame and communicate it to the application.

### *IDL*

---

[id(8), helpstring("method DeviceInfoEvent")] HRESULT DeviceInfoEvent(short DevID);

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

DevID as short with the ID of the device becoming active

### *EXAMPLE*

---

```
...
Private Sub RFNCCTL_DeviceInfoEvent(ByVal DevID As Integer)
Dim db As Database
Dim rs As Recordset
Dim strAnser As String
Dim strSQL As String

    strSQL = "SELECT * FROM DeviceTable WHERE DevID = '" & CStr(DevID)
    & "'"
    Set db = OpenDatabase("C:\Example\DeviceDB.mdb")
    Set rs = db.OpenRecordset(strSQL, , dbOpenDynaset)
    On Error GoTo ErrorHandler
    '
    ' We have found the device so we do not add it to DB
    '
    rs.MoveFirst
    db.Close
    Exit Sub

ErrorHandler:
    rs.AddNew
    rs.Fields("DevID") = CStr(DevID)
    rs.Fields("Answer") = "Answer for device " & CStr(DevID)
    rs.Update
End Sub
...
```

## *EVENT **ERROR** EVENT*

### *PURPOSE*

---

An error has occurred.

### *DESCRIPTION*

---

This event notifies the container when an error has occurred.

### *IDL*

---

```
[id(1), helpstring("method ErrorEvent")] HRESULT ErrorEvent(short ErrorID, BSTR ErrorDesc);
```

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

ErrorID as short with the ID of the error occurred  
ErrorDesc as String containing the error description

### *EXAMPLE*

---

```
...  
Private Sub RFNCCTL_ErrorEvent(ByVal ErrorID As Integer, ByVal ErrorDesc As String)  
    MsgBox ErrorDesc, vbCritical, "Error n. " & CStr(ErrorID)  
End Sub...
```

## *EVENT **LOSTFRAMEEVENT***

### *PURPOSE*

---

Device lost frame.

### *DESCRIPTION*

---

This event will be sent from the control to the application when the PC notices that at least one frame of a device has been lost. See **SetLostFrameBehaviour** method on page 34 for details.

### *IDL*

---

```
[id(7), helpstring("method LostFrameEvent")] HRESULT LostFrameEvent(short DevID,
long NumLostFrame)
```

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

DevID as short with the ID of the device which has lost a frame  
 NumLostFrame as long representing the number of lost frames

### *EXAMPLE*

---

```
...
Private Sub RFNCCTL_LostFrameEvent(ByVal DevID As Integer,
ByVal NumLostFrame As Long)
On Error Resume Next
Dim StrMsg as String
    StrMsg = "Lost Frame on Device : " & CStr(DevID) & _
        Chr$(10)
    StrMsg = StrMsg & "Lost frame number : " & CStr(Frame)
    MsgBox StrMsg & , vbExclamation
End Sub
...
```

**EVENT SATCFGEVENT****PURPOSE**

Base station configuration answer.

**DESCRIPTION**

This event will be sent from the control to the application when a base station answers to a **GetConfiguration** command. Before sending this event the control sets all the related properties.

**IDL**

[id(4), helpstring("method SatCfgEvent")] HRESULT SatCfgEvent(short SatID);

**INPUT PARAMETERS**

None

**OUTPUT VALUE**

SatID as short with the ID of the base station sending the configuration answer.

**EXAMPLE**

```

...
Private Sub RFNCCTL_SatCfgEvent(ByVal SatID As Integer)
On Error GoTo ErrCfgGet
Dim SatObject as Sat
Dim StrProperties as String
,
, ' Show the base station properties
,
Set SatObject = RFNCCTL.Satellites.Satellite(SatID)
StrProperties = strProperties & "SatID = "& _
CStr(SatObject.SatID) & Chr(10)
StrProperties = strProperties & "PID = " & _
CStr(SatObject.PID) & Chr(10)
StrProperties = strProperties & "RF Version = " & _
CStr(SatObject.RFMaxProtocolVersion) & "." & _
CStr(SatObject.RFMinProtocolVersion) & Chr(10)
MsgBox StrProperties, vbExclamation, "Satellite n. " & _
CStr(SatID)
Exit Sub
ErrCfgGet:
MsgBox "Error SatCfgGet Event", vbCritical, "Test Program"
End Sub
...

```

## *EVENT **SLEEP**SAT**EVENT***

### *PURPOSE*

---

Base station sleeping.

### *DESCRIPTION*

---

This event will be sent from the control to the application when a base station does not answer to the data collection requested by the PC for a certain number of times. This means the base station might be broken or the serial line (RS 232 or RS 485) might break off.

### *IDL*

---

[id(6), helpstring("method SleepSatEvent")] HRESULT SleepSatEvent(short SatID, short Status);

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

SatID as short with the ID of the base station not answering

Status as short representing the base station status. The possible values of these parameters are SAT\_ON = 1, SAT\_OFF = 2 or SAT\_SLEEP = 3.

---

**EXAMPLE**

---

```
...
Private Const SAT_ON = 1
Private Const SAT_OFF = 2
Private Const SAT_SLEEP = 3
...
Private Sub RFNCCTL_SleepSatEvent(ByVal SatID As Integer, ByVal
Status As Integer)
On Error GoTo ErrStatus
Dim strMsg As String
    strMsg = ""
    Select Case Status
        Case SAT_ON
            strMsg = "Satellite " & CStr(SatID) & " is ON"
        Case SAT_OFF
            strMsg = "Satellite " & CStr(SatID) & " is OFF"
        Case SAT_SLEEP
            strMsg = "Satellite " & CStr(SatID) & " is SLEEP"
    End Select
    MsgBox strMsg, vbExclamation
    Exit Sub
ErrStatus:
    MsgBox "Error Sleep Sat Event", vbCritical, "Test Program"
End Sub
...
```

## *EVENT STATUS EVENT*

### *PURPOSE*

---

Base station status answer.

### *DESCRIPTION*

---

This event will be sent from the control to the application when a base station answers to a **GetStatus** command. Before sending this event, the control sets all the related properties.

### *IDL*

---

[id(5), helpstring("method StatusEvent")] HRESULT StatusEvent(short SatID);

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

SatID as short with the ID of the base station sending the Get Status answer

### *EXAMPLE*

---

```
...
Private Sub RFNCCTL_StatusEvent(ByVal SatID As Integer)
On Error GoTo ErrStatus
Dim SatObject As Sat
Dim strMsg As String
    Set SatObject = RFNCCTL.Satellites.Satellite(SatID)
    StrMsg = ""
    strMsg = strMsg & "Rx Free   : " & sSat.RxFree & Chr(10)
    strMsg = strMsg & "Tx Free   : " & sSat.TxFree & Chr(10)
    strMsg = strMsg & "Rx Total  : " & sSat.RxTotal & Chr(10)
    strMsg = strMsg & "Tx Total  : " & sSat.TxTotal
    MsgBox strMsg, vbExclamation, "Satellite " & CStr(SatID) &
        " status .."
    Exit Sub
ErrStatus:
    MsgBox "Error Status Event", vbCritical, "Test Program"
End Sub
...
```

## 4 BASE STATION COLLECTION OBJECT

---

### 4.1 PROPERTIES

#### *PROPERTY COUNT*

##### *PURPOSE*

---

Gets the total number of base stations.

##### *DESCRIPTION*

---

This property gives the number of base stations stored in the collection.

##### *IDL*

---

[propget, id(3), helpstring("property Count")] HRESULT Count([out, retval] long \*pVal)

##### *DATA TYPE*

---

LONG

##### *POSSIBLE VALUES*

---

Read only property

##### *EXAMPLE*

---

```
...  
Dim SatellitesObject as Sats  
Dim NumSats as Long  
...  
Set SatelliteObject = RFNCCTL.Satellites  
NumSats = SatelliteObject.Count  
...  
NumSats = RFNCCTL.Satellites.Count  
..
```

## PROPERTY **ITEM**

### PURPOSE

---

Gets a base station object from the collection.

### DESCRIPTION

---

This property retrieves a base station object from the collection. You must set an index representing the offset in the array used to store all the base station objects (the array is 0-based).

### IDL

---

```
[propget, id(DISPID_VALUE)] HRESULT Item([in]long Index, [out, retval] VARIANT *pVal)
```

### INPUT PARAMETERS

---

Index : Long

The index of the array element used to store the collection of base stations.

### DATA TYPE

---

Pointer to dispatch the object interface.

### POSSIBLE VALUES

---

Read only property.

### EXAMPLE

---

```
...
Dim SatellitesObject as Sats
Dim i as Long
...
Set SatellitesObjects = RFNCCTL.Satellites
For i = 0 To SatellitesObject.Count - 1
    MsgBox "Satellite n. " & CStr(i) & " ID "& _
        CStr(Satellites.Item(i).SatID)
Next i
...
```

*PROPERTY **SATELLITE****PURPOSE*

Gets a base station object from the collection.

*DESCRIPTION*

This property retrieves a base station object from the collection. You must set the SatID of the desired base station.

*IDL*

```
[propget, id(4)] HRESULT Satellite([in]VARIANT SatID, [out, retval] ISat **pVal);
```

*INPUT PARAMETERS*

SatID : Short  
The ID of the desired base station object.

*DATA TYPE*

Pointer to dispatch the object interface.

*POSSIBLE VALUES*

Read only property

*EXAMPLE*

```
...
Dim SatellitesObject as Sats
Dim SatObject as Sat
...
Set SatellitesObjects = RFNCCTL.Satellites
SatellitesObjects.Add(1500)
...
Set SatObject = SatellitesObjects.Satellite(1500)
...
```

## 4.2 METHODS

### *METHOD **Add***

#### *PURPOSE*

---

Adds a base station object.

#### *DESCRIPTION*

---

This method adds a base station object to the collection. You can add base stations with their ID between 0 and 1999.

#### *IDL*

---

```
[id(1), helpstring("method Add")] HRESULT Add([in]VARIANT SatID, [out, retval] ISat  
**pVal);
```

#### *INPUT PARAMETERS*

---

SatID: Short

The ID of the base station object to be added.

#### *OUTPUT VALUE*

---

Pointer to dispatch the interface of the object just added.

#### *EXAMPLE*

---

```
...  
Dim SatObject as Sat  
...  
' Add base station 1500 to the collection  
Set SatObject = RFNCCTL.Satellite.Add(1500)  
SatObject.GetConfiguration  
...
```

---

**METHOD *CLEAR***

---

**PURPOSE**

Clears the collection.

---

**DESCRIPTION**

This method removes all the base stations from the collection.

---

**IDL**

[id(5), helpstring("method Clear")] HRESULT Clear();

---

**INPUT PARAMETERS**

None

---

**OUTPUT VALUE**

None

---

**EXAMPLE**

---

```
...  
' Remove all base stations from the collection  
RFNCCTL.Satellites.Clear  
...
```

## *METHOD REMOVE*

### *PURPOSE*

---

Removes a base station object.

### *DESCRIPTION*

---

This method removes a base station object from the collection. If the base station does not exist, the function fails.

### *IDL*

---

```
[id(2), helpstring("method Remove")] HRESULT Remove([in]VARIANT SatID);
```

### *INPUT PARAMETERS*

---

SatID: Short

The ID of base station object to be removed.

### *OUTPUT VALUE*

---

None

### *EXAMPLE*

---

```
...  
Dim SatObject as Sat  
...  
Set SatObject = RFNCCTL.Satellites.Add(1500)  
SatObject.GetConfiguration  
...  
' Remove base station 1500 from the collection  
RFNCCTL.Satellites.Remove(1500)  
...
```

## 5 DEVICE COLLECTION OBJECT

---

### 5.1 PROPERTIES

#### *PROPERTY COUNT*

##### *PURPOSE*

---

Gets the total number of devices.

##### *DESCRIPTION*

---

This property gets the number of devices stored in the collection.

##### *IDL*

---

[propget, id(3), helpstring("property Count")] HRESULT Count([out, retval] long \*pVal);

##### *DATA TYPE*

---

LONG

##### *POSSIBLE VALUES*

---

Read only property

##### *EXAMPLE*

---

```
...  
Dim DevicesObject as Devices  
Dim NumDevs as Long  
...  
Set DevicesObject = RFNCCTL.Devices  
NumDevs = DevicesObject.Count  
...  
NumDevs = RFNCCTL.Devices.Count  
...
```

## *PROPERTY **DATALen***

### *PURPOSE*

---

Gets the length in bytes of data set in DeviceData property.

### *DESCRIPTION*

---

This property may be only read and provides the length in bytes of data sent from a device. **This property is allowed only when working in the DataEvent handler.**

### *IDL*

---

```
[propget, id(9), helpstring("property DataLen")] HRESULT DataLen([out, retval]  
short*pVal);
```

### *DATA TYPE*

---

Short

### *POSSIBLE VALUES*

---

1 up to 4095

**EXAMPLE**

```

...
Private Sub RFNCCTL_DataEvent(ByVal SatID As Integer, ByVal
DevID As Integer, ByVal Kind As Boolean)
On Error Resume Next
Dim item As ListItem
Dim sDevice As Device
Dim strAnswer As String
Dim vaData As Variant
Dim strData As String
Dim byteData As Byte
Dim i As Integer

    Set sDevice = OCX.Devices.Device(DevID)
    vaData = sDevice.DeviceData
    '
    'Data interpretation
    '
    strData = ""
    If OCX.Mode <> MODE_BINARY Then
        'vaData has been set to VT_BSTR type by the control
        strData = CStr(vaData)
    Else
        'vaData has been set to VT_ARRAY | VT_UI1 type by the
control
        For i = 0 To sDevice.DataLen - 1
            byteData = vaData(i)
            strData = strData & Right("000" + CStr(byteData),3)+""
        Next i
    End if
    Set item = ListData.ListItems.Add(, , CStr(nItem))
    item.ListSubItems.Add , , CStr(DevID)
    item.ListSubItems.Add , , strData

...
End Sub

```

## *PROPERTY **DEVICE***

### *PURPOSE*

---

Gets a device object from the collection.

### *DESCRIPTION*

---

This property gets a device object from the collection. You must set the DevID of the desired device.

### *IDL*

---

```
[propget, id(4)] HRESULT Device([in]VARIANT DevID, [out, retval] IDevice **pVal);
```

### *INPUT PARAMETERS*

---

SatID : Short

The ID of desired device object.

### *DATA TYPE*

---

Pointer to dispatch the object interface.

### *POSSIBLE VALUES*

---

Read only property

### *EXAMPLE*

---

```
...  
Dim DevicesObject as Devices  
Dim DevObject as Device  
...  
Set DevicesObjects = RFNCCTL.Devices  
DevicesObjects.Add(1500)  
...  
Set DevObject = DevicesObjects.Device(1500)  
...
```

## *PROPERTY* **DEVICE DATA**

### *PURPOSE*

---

Gets data from a device.

### *DESCRIPTION*

---

This property may be only read and represents data received from a device. **This property is allowed only when working in the DataEvent handler.**

### *IDL*

---

```
[propget, id(8), helpstring("property DeviceData")] HRESULT DeviceData([out, retval]VARIANT *pVal);
```

### *DATA TYPE*

---

VARIANT

### *POSSIBLE VALUES*

---

VT\_BSTR if Mode is 0,2,3,4 (Text Mode)  
VT\_ARRAY | VT\_UI1 if Mode is 1 (Binary Mode)

**EXAMPLE**

---

```

...
Private Sub RFNCCTL_DataEvent(ByVal SatID As Integer, ByVal
DevID As Integer, ByVal Kind As Boolean)
On Error Resume Next
Dim item As ListItem
Dim sDevice As Device
Dim strAnswer As String
Dim vaData As Variant
Dim strData As String
Dim byteData As Byte
Dim i As Integer

    Set sDevice = OCX.Devices.Device(DevID)
    vaData = sDevice.DeviceData
    '
    'Data interpretation
    '
    strData = ""
    If OCX.Mode <> MODE_BINARY Then
        'vaData has been set to VT_BSTR type by the control
        strData = CStr(vaData)
    Else
        'vaData has been set to VT_ARRAY | VT_UI1 type by the
control
        For i = 0 To sDevice.DataLen - 1
            byteData = vaData(i)
            strData = strData & Right("000" + CStr(byteData),3)+""
        Next i
    End if
    Set item = ListData.ListItems.Add(, ,CStr(nItem))
    item.ListSubItems.Add , , CStr(DevID)
    item.ListSubItems.Add , , strData

...
End Sub

```

## *PROPERTY ITEM*

### *PURPOSE*

---

Gets a device object from the collection.

### *DESCRIPTION*

---

This property gets a device object from the collection. You must set an index representing the offset in the array used to store all the device objects (the array is 0-based).

### *IDL*

---

```
[propget, id(DISPID_VALUE)] HRESULT Item([in]long Index, [out, retval] VARIANT *pVal);
```

### *INPUT PARAMETERS*

---

Index : Long

The index of the array element used to store the collection of devices.

### *DATA TYPE*

---

Pointer to dispatch the object interface.

### *POSSIBLE VALUES*

---

Read only property

### *EXAMPLE*

---

```
...
Dim DevicesObject as Devices
Dim i as Long
...
Set DevicesObjects = RFNCCTL.Devices
For i = 0 To DevicesObject.Count - 1
    MsgBox "Device n. " & CStr(i) & " ID "& _
        CStr(Depices.Item(i).DevID)
Next i
...
```

## 5.2 METHODS

### *METHOD **Add***

#### *PURPOSE*

---

Adds a device object.

#### *DESCRIPTION*

---

This method adds a device object to the collection. You may add devices with their ID between 0 and 1999.

#### *IDL*

---

```
[id(1), helpstring("method Add")] HRESULT Add(VARIANT DevID, [out, retval]  
IDevice **pVal)
```

#### *INPUT PARAMETERS*

---

DevID: Short  
The ID of device object to be added.

#### *OUTPUT VALUE*

---

Pointer to dispatch the interface of the object just added.

#### *EXAMPLE*

---

```
...  
Dim DevObject as Device  
...  
' Add device 1500 to the collection  
Set DevObject = RFNCCTL.Devices.Add(1500)  
DevObject.DataWrite "foo string data"  
...
```

---

**METHOD *CLEAR***

---

**PURPOSE**

Clears the collection.

---

**DESCRIPTION**

This method removes all the devices from the collection.

---

**IDL**

[id(5), helpstring("method Clear")] HRESULT Clear();

---

**INPUT PARAMETERS**

None

---

**OUTPUT VALUE**

None

---

**EXAMPLE**

---

```
...  
' Remove all devices from the collection  
RFNCCTL.Devices.Clear  
...
```

## *METHOD REMOVE*

### *PURPOSE*

---

Removes a device object.

### *DESCRIPTION*

---

This method removes a device object from the collection. If the device does not exist, the function fails.

### *IDL*

---

[id(2), helpstring("method Remove")] HRESULT Remove([in]VARIANT SatID);

### *INPUT PARAMETERS*

---

SatID: Short

The ID of device object to be removed.

### *OUTPUT VALUE*

---

None

### *EXAMPLE*

---

```
...  
Dim DevObject as Sat  
...  
Set DevObject = RFNCCTL.Device.Add(1500)  
DevObject.DataWrite "foo string data"  
...  
' Remove device 1500 from the collection  
RFNCCTL.Devices.Remove(1500)  
...
```

## 6 BASE STATION OBJECT

---

### 6.1 PROPERTIES

#### *PROPERTY* **BAUDRATERF**

##### *PURPOSE*

---

Indicates the radio baud rate.

##### *DESCRIPTION*

---

This property sets the radio baud rate value between the base station and the device.

##### *IDL*

---

```
[propget, id(7), helpstring("property BaudRateRF")] HRESULT BaudRateRF([out,
retval] long *pVal);
[propput, id(7), helpstring("property BaudRateRF")] HRESULT BaudRateRF([in] long
newVal);
```

##### *DATA TYPE*

---

LONG

##### *POSSIBLE VALUES*

---

9600; 19200 (for European base station models)  
fixed 36800 (for USA base station models)

##### *EXAMPLE*

---

```
...
Dim SatObject as Sat
Dim strProperties as String
...
' Show the base station BaudRateRF property
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
strProperties = "BaudRateRF = " & CStr(SatObject.BaudRateRF)
...
```

## PROPERTY **BAUDRATES**

### PURPOSE

---

Indicates the serial communication baud rate.

### DESCRIPTION

---

This property sets the RS232 and RS485 baud rate between the base station and the host PC.

### IDL

---

```
[propget, id(6), helpstring("property BaudRateRS")] HRESULT BaudRateRS([out,
retval] long *pVal);
[propput, id(6), helpstring("property BaudRateRS")] HRESULT BaudRateRS([in] long
newVal);
```

### DATA TYPE

---

LONG

### POSSIBLE VALUES

---

300; 600; 1200; 2400; 9600; 19200; 38400; 57600; 115200

### EXAMPLE

---

```
...
Dim SatObject as Sat
Dim strProperties as String
...
' Show the base station BaudRateRS property
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
strProperties = "BaudRateRS = " & CStr(SatObject.BaudRateRS)
...
```

## PROPERTY **BUFFERSIZE**

This property is reserved.

## PROPERTY **BUFFERTHRESHOLD**

This property is reserved.

**PROPERTY *DIRECT***

This property is reserved.

**PROPERTY *ENABLED*****PURPOSE**

---

Enables/disables the base station.

**DESCRIPTION**

---

This property enables or disables the base station. When the base station is disabled, it enters a particular state (OFF state) and is not polled by the network control thread.

This property is useful during the network data collection, since at polling start up all base stations should be already enabled (Enable = TRUE).

**IDL**

---

```
[propget, id(2), helpstring("property Enabled")] HRESULT Enabled([out, retval] BOOL *pVal);  
[propput, id(2), helpstring("property Enabled")] HRESULT Enabled([in] BOOL newVal);
```

**DATA TYPE**

---

BOOL

**POSSIBLE VALUES**

---

The control accepts only these values:

TRUE, FALSE

The default value is TRUE.

**EXAMPLE**

---

```
...  
Dim SatObject as Sat  
...  
'Set the base station to Off state  
Set SatObject = RFNCCTL.Satellites.Satellite(1500)  
SatObject.Enabled = False  
...
```

*PROPERTY FRAMELIFE*

This property is reserved.

*PROPERTY INTERFACE**PURPOSE*

---

Indicates the base station function mode.

*DESCRIPTION*

---

This property sets the RS232 or RS485 interface. If it is set to 0 the RS485 interface is selected; while the 1 value selects the RS232 one.

*IDL*

---

```
[propget, id(12), helpstring("property Interface")] HRESULT Interface([out, retval];
short *pVal);
[propput, id(12), helpstring("property Interface")] HRESULT Interface([in] short newVal);
```

*DATA TYPE*

---

SHORT

*POSSIBLE VALUES*

---

0; 1

*EXAMPLE*

---

```
...
Dim SatObject as Sat
Dim strProperties as String
...
' Show the base station interface property
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
strProperties = "Interface = " & CStr (SatObject.Interface)
...
```

**PROPERTY *MAXBACKOFF***

This property is reserved.

**PROPERTY *MINBACKOFF***

This property is reserved.

**PROPERTY *PID*****PURPOSE**

Product ID

**DESCRIPTION**

This property is set from the DLL network manager and represents an ID of the product. Each Datalogic RF product has a different ID.

**IDL**

```
[propget, id(3), helpstring("property PID")] HRESULT PID([out, retval] short *pVal);
[propput, id(3), helpstring("property PID")] HRESULT PID([in] short newVal);
```

**DATA TYPE**

SHORT

**POSSIBLE VALUES**

The STARGATE™ RF Base Station PID value is 10.  
The OM-DRAGON M101 Radio Cradle PID value is 19729.

**EXAMPLE**

```
...
Dim SatObject as Sat
Dim strProperties as String
...
' Show the base station properties
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
strProperties = strProperties & "SatID = "& _
CStr(SatObject.SatID)
strProperties = strProperties & "PID = " & CStr(SatObject.PID)
strProperties = strProperties & "RF Version = " & _
CStr(SatObject.RFMaxProtocolVersion) & "." & _
CStr(SatObject.RFMinProtocolVersion)
...
```

*PROPERTY **POWERSAVE***

This property is reserved.

*PROPERTY **PREAMBLELENGTH***

This property is reserved.

*PROPERTY **RFMaxPROTOCOLVERSION****PURPOSE*

---

RF Version.

*DESCRIPTION*

---

This property is set from the DLL network manager and represents the major version of the RF protocol.

*IDL*

---

```
[propget, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT  
RFMaxProtocolVersion([out, retval] short *pVal);  
[propput, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT  
RFMaxProtocolVersion([in] short newVal);
```

*DATA TYPE*

---

SHORT

*POSSIBLE VALUES*

---

Greater than 0

---

*EXAMPLE*

---

```
...
Dim SatObject as Sat
Dim strProperties as String
...
' Show the base station properties
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
StrProperties = strProperties & "SatID = "& _
CStr(SatObject.SatID)
StrProperties = strProperties & "PID = " & CStr(SatObject.PID)
StrProperties = strProperties & "RF Version = " & _
CStr(SatObject.RFMaxProtocolVersion) & ". "& _
CStr(SatObject.RFMinProtocolVersion)
...
```

**PROPERTY *RFMinProtocolVersion*****PURPOSE**

RF Version.

**DESCRIPTION**

This property is set from the DLL network manager and represents the minor version of the RF protocol.

**IDL**

```
[propget, id(5), helpstring("property RFMinProtocolVersion")] HRESULT
RFMinProtocolVersion([out, retval] short *pVal);
[propput, id(5), helpstring("property RFMinProtocolVersion")] HRESULT
RFMinProtocolVersion([in] short newVal);
```

**DATA TYPE**

SHORT

**POSSIBLE VALUES**

Greater than 0

**EXAMPLE**

```
...
Dim SatObject as Sat
Dim strProperties as String
...
' Show the base station properties
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
strProperties = strProperties & "SatID = "& _
CStr(SatObject.SatID)
strProperties = strProperties & "PID = " & CStr(SatObject.PID)
strProperties = strProperties & "RF Version = " & _
CStr(SatObject.RFMaxProtocolVersion) & "." & _
CStr(SatObject.RFMinProtocolVersion)
...
```

**PROPERTY *RxFree*, *RxTotal*, *TxFree*, *TxTotal*****PURPOSE**

---

Gets the size of free receiving buffer on STARGATE™.

**DESCRIPTION**

---

This property is set from the DLL network manager with data sent by STARGATE™.

**IDL**

---

```
[propget, id(17), helpstring("property RxFree")] HRESULT RxFree([out, retval]  
short *pVal);  
[propget, id(18), helpstring("property RxTotal")] HRESULT RxTotal([out, retval] short*  
pVal);  
[propget, id(19), helpstring("property TxFree")] HRESULT TxFree([out, retval]  
short *pVal);  
[propget, id(20), helpstring("property TxTotal")] HRESULT TxTotal([out, retval]  
short *pVal);
```

**DATA TYPE**

---

SHORT

**POSSIBLE VALUES**

---

Greater than 0

*EXAMPLE*

---

```

...
Dim SatObject as Sat
...
' Get base station configuration
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
SatObject.GetStatus
...
' Get Status Event Handler
,
Private Sub RFNCCTL_StatusEvent(ByVal SatID As Integer)
On Error GoTo ErrStatus
Dim sSat As Sat
Dim strMsg As String
Set sSat = RFNCCTL.Satellites.Satellite(SatID)
strMsg = ""
strMsg = strMsg & "Rx Free   : " & sSat.RxFree & Chr(10)
strMsg = strMsg & "Tx Free   : " & sSat.TxFree & Chr(10)
strMsg = strMsg & "Rx Total  : " & sSat.RxTotal & Chr(10)
strMsg = strMsg & "Tx Total  : " & sSat.TxTotal
MsgBox strMsg, vbExclamation, "Satellite " & CStr(SatID) &
" status .."
Exit Sub
ErrStatus:
MsgBox "Error Status Event", vbCritical, "Test Program"
End Sub

```

---

*PROPERTY **SatID***

---

*PURPOSE*

Gets the base station ID.

---

*DESCRIPTION*

This property gets and sets the base station ID. It is very important, since the network data collection involves only the base stations having an ID.

---

*IDL*

---

```
[propget, id(1), helpstring("property SatID")] HRESULT SatID([out, retval] short *pVal);  
[propput, id(1), helpstring("property SatID")] HRESULT SatID([in] short newVal);
```

---

*DATA TYPE*

SHORT

---

*POSSIBLE VALUES*

0 .. 1999

---

*EXAMPLE*

---

```
...  
Dim SatObject as Sat  
Dim satAddr as Integer  
Set SatObject = RFNCCTL.Satellites.Add(1500)  
...  
' Get the Satellite ID  
satAddr = SatObject.SatID  
...
```

## PROPERTY **NewSatID**

### PURPOSE

---

Indicates the new ID to be assigned to a base station.

### DESCRIPTION

---

This property sets the new ID to assign to the base station. This ID will be active after calling the SetConfiguration() method.

### IDL

---

```
[propget, id(24), helpstring("property NewSatID")] HRESULT NewSatID([out, retval]
short *pVal);
[propput, id(24), helpstring("property NewSatID")] HRESULT NewSatID([in] short
newVal);
```

### DATA TYPE

---

SHORT

### POSSIBLE VALUES

---

0 .. 1999

### EXAMPLE

---

```
...
Dim SatObject as Sat
Set SatObject = RFNCCTL.Satellites.Add(1500)
...
' Sets the Satellite ID
SatObject.NewSatID = 1505
SatObject.SetConfiguration
...
```

## 6.2 METHODS

### *METHOD **ALIVE***

#### *PURPOSE*

---

Calls a base station.

#### *DESCRIPTION*

---

This method calls a base station to know if it is alive or not. When sending an alive command to a base station present and operating in the network, it will answer with a particular string. This command is totally asynchronous and managed through a secondary thread which causes the STAR-Link™ ActiveX to answer with an event called **AliveEvent** containing the base station string information. If the Alive command is not right, the control sends an error event.

#### *IDL*

---

```
[id(21), helpstring("method Alive")] HRESULT Alive();
```

#### *INPUT PARAMETERS*

---

None

#### *OUTPUT VALUE*

---

None

#### *POSSIBLE VALUES*

---

None

#### *EXAMPLE*

---

```
...
Dim SatObject as Sat
...
'Asking if base station is Alive
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
SatObject.Alive
...
;...
' Alive Event Handler
,
Private Sub RFNCCTL_AliveEvent(ByVal SatID As Integer, ByVal
Info As String)
    MsgBox Info, vbExclamation, "Satellite : " & CStr(SatID)
End Sub
```

## *METHOD* **GETCONFIGURATION**

### *PURPOSE*

---

Gets a base station configuration.

### *DESCRIPTION*

---

This method asks for the base station configuration. When sending this command to an alive base station, it will answer with its configuration based on the related property. This command (like **Alive** command and next command) is totally asynchronous, managed through a secondary thread which causes the STAR-Link™ ActiveX to answer with an event called **SatCfgEvent**. It contains the ID of the base station answering to the command. Before sending the event, the control sets all the base station properties.

If the **GetConfiguration** command is not right, the control sends an error event (**ErrorEvent**).

### *IDL*

---

```
[id(22), helpstring("method GetConfiguration")] HRESULT GetConfiguration();
```

### *INPUT PARAMETERS*

---

None

### *OUTPUT VALUE*

---

None

### *POSSIBLE VALUES*

---

None

---

*EXAMPLE*

---

```
...
Dim SatObject as Sat
'...
' Get base station configuration
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
SatObject.GetConfiguration
'...
' Get Configuration Event Handler
,
Private Sub RFNCCTL_ SatCfgEvent(ByVal SatID As Integer)
On Error Resume Next
Dim StrCfg as String
Dim SatelliteObject as Sat
    StrCfg = ""
    StrCfg = StrCfg & "Satellite ID " & CStr(SatID) & Chr$(10)
    Set SatObject = RFNCCTL.Satellites.Satellite(SatID)
    StrCfg = StrCfg & "PID" & CStr(SatObject.PID) & Chr$(10)
    StrCfg = StrCfg & "RF Version " &
CStr(SatObject.RFMaxProtocolVersion) & "." &
CStr(SatObject.RFMinProtocolVersion) & Chr$(10)

    MsgBox StrCfg, vbExclamation, "Satellite Parameters"
End Sub
```

## METHOD **GetStatus**

### PURPOSE

---

Gets the base station status.

### DESCRIPTION

---

This method asks for the base station input/output buffer state. If this command is sent to an alive base station, it will answer by setting the four properties **RxFree**, **TxFree**, **RxTotal** and **TxTotal** containing the number of bytes free and the total number of the two base station buffers. This command is totally asynchronous (like the others) and the STAR-Link™ ActiveX answers with an event called **StatusEvent**. Before sending this event, the control sets all the base station properties. If the **GetStatus** command is not right, the control sends an error event (**ErrorEvent**).

### IDL

---

```
[id(22), helpstring("method GetConfiguration")] HRESULT GetConfiguration();
```

### INPUT PARAMETERS

---

None

### OUTPUT VALUE

---

None

*EXAMPLE*

```

...
Dim SatObject as Sat
...
' Get base station configuration
Set SatObject = RFNCCTL.Satellites.Satellite(1500)
SatObject.GetStatus
...
' Get Status Event Handler
,
Private Sub RFNCCTL_StatusEvent(ByVal SatID As Integer)
On Error GoTo ErrStatus
Dim sSat As Sat
Dim strMsg As String
Set sSat = RFNCCTL.Satellites.Satellite(SatID)
strMsg = ""
strMsg = strMsg & "Rx Free   : " & sSat.RxFree & Chr(10)
strMsg = strMsg & "Tx Free   : " & sSat.TxFree & Chr(10)
strMsg = strMsg & "Rx Total  : " & sSat.RxTotal & Chr(10)
strMsg = strMsg & "Tx Total  : " & sSat.TxTotal
MsgBox strMsg, vbExclamation, "Satellite " & CStr(SatID) &
" status .."
Exit Sub
ErrStatus:
MsgBox "Error Status Event", vbCritical, "Test Program"
End Sub

```

## METHOD **SetConfiguration**

### PURPOSE

---

Sets the base station configuration.

### DESCRIPTION

---

This method sets new configuration parameters. **Be careful while setting the parameters because a wrong configuration causes an incorrect operation of the base station.**

Since the SetConfiguration method updates all the base station parameters, it is advised to follow the given procedure to change or update **some** of the base station properties:

1. select the GetConfiguration method to get the current value of all the base station properties;
2. change the desired properties only;
3. select the SetConfiguration method.

If the **SetConfiguration** command is not right, the control sends an error event (**ErrorEvent**).

### IDL

---

```
[id(23), helpstring("method SetConfiguration")] HRESULT SetConfiguration();
```

### INPUT PARAMETERS

---

None

### OUTPUT VALUE

---

None

### POSSIBLE VALUES

---

None

### EXAMPLE

---

```
...  
Dim SatObject as Sat  
...  
' Setting new ID to the satellite  
Set SatObject = RFNCCTL.Satellites.Satellite(1500)  
SatObject.NewSatID = 1501  
SatObject.SetConfiguration  
...
```

## 7 DEVICE OBJECT

---

### 7.1 PROPERTIES

#### *PROPERTY* **BUFFERSIZE**

This property is reserved.

#### *PROPERTY* **DevID**

##### *PURPOSE*

---

Gets or sets the device ID.

##### *DESCRIPTION*

---

This property gets or sets the device ID.

##### *IDL*

---

```
[propget, id(1), helpstring("property DevID")] HRESULT DevID([out, retval] short *pVal);  
[propput, id(1), helpstring("property DevID")] HRESULT DevID([in] short newVal);
```

##### *DATA TYPE*

---

SHORT

##### *POSSIBLE VALUES*

---

Greater than 1

##### *EXAMPLE*

---

```
...  
Dim DevObject as Device  
Set DevObject = RFNCCTL.Devices.Add(1500)  
...  
' Change the Device ID  
DevObject.DevID = 1501  
...
```

## *PROPERTY **ENABLED***

### *PURPOSE*

---

Enables or disables the device.

### *DESCRIPTION*

---

By setting off a device, it does not send any data event to the container. The network thread keeps on receiving the device data, but it does not process it.

### *IDL*

---

```
[propget, id(2), helpstring("property Enabled")] HRESULT Enabled([out, retval] BOOL *pVal);  
[propput, id(2), helpstring("property Enabled")] HRESULT Enabled([in] BOOL newVal);
```

### *DATA TYPE*

---

BOOL

### *POSSIBLE VALUES*

---

The control accepts only these values:

TRUE, FALSE

The default value is TRUE.

### *EXAMPLE*

---

```
...  
Dim DevObject as Device  
...  
' Set the device to Off state  
Set DevObject = RFNCCTL.Devices.Device(1500)  
DevObject.Enabled = False  
...
```

**PROPERTY *PID*****PURPOSE**

Product ID.

**DESCRIPTION**

This property is set from the DLL network manager and represents an ID of the product. Each Datalogic RF product has a different ID.

**IDL**

```
[propget, id(6), helpstring("property PID")] HRESULT PID([out, retval] short *pVal);
[propput, id(6), helpstring("property PID")] HRESULT PID([in] short newVal);
```

**DATA TYPE**

SHORT

**POSSIBLE VALUES**

The PID value of Datalogic RF products is listed in the following table:

| Product        | PID Value |
|----------------|-----------|
| Undefined      | 0         |
| Formula 734/RF | 1         |
| Formula 725/RF | 2         |
| Formula 660/RF | 3         |
| Gryphon™ M100  | 19761     |
| Gryphon™ M200  | 19762     |
| Dragon M101    | 20241     |
| OM-Dragon M101 | 19729     |
| STAR-Modem     | 20300     |

**EXAMPLE**

```
...
Dim DevObject as Device
Dim strProperties as String
...
' Show the Device properties
Set DevObject = RFNCCTL.Devices.Device(1500)
strProperties = strProperties & "DevID = "& _
CStr(DevObject.DevID)
strProperties = strProperties & "PID = " & CStr(DevObject.PID)
strProperties = strProperties & "RF Version = " & _
CStr(DevObject.RFMaxProtocolVersion) & "." & _
CStr(DevObject.RFMinProtocolVersion)
...
```

**PROPERTY *RFMaxProtocolVersion*****PURPOSE**

RF Version.

**DESCRIPTION**

This property is set from the DLL network manager and represents the major version of the RF protocol.

**IDL**

```
[propget, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT
RFMaxProtocolVersion([out, retval] short *pVal);
[propput, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT
RFMaxProtocolVersion([in] short newVal);
```

**DATA TYPE**

SHORT

**POSSIBLE VALUES**

Greater than 1

**EXAMPLE**

```
...
Dim DevObject as Device
Dim strProperties as String
...
' Show the Device properties
Set DevObject = RFNCCTL.Devices.Device(1500)
strProperties = strProperties & "DevID = "& _
CStr(DevObject.DevID)
strProperties = strProperties & "PID = " & CStr(DevObject.PID)
strProperties = strProperties & "RF Version = " & _
CStr(DevObject.RFMaxProtocolVersion) & "." & _
CStr(DevObject.RFMinProtocolVersion)
...
```

**PROPERTY *RFMinProtocolVersion*****PURPOSE**

RF Version.

**DESCRIPTION**

This property is set from the DLL and represents the minor version of the RF protocol.

**IDL**

```
[propget, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT
RFMaxProtocolVersion([out, retval] short *pVal);
[propput, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT
RFMaxProtocolVersion([in] short newVal);
```

**DATA TYPE**

SHORT

**POSSIBLE VALUES**

Greater than 1

**EXAMPLE**

```
...
Dim DevObject as Device
Dim strProperties as String
'...
' Show the Device properties
Set DevObject = RFNCCTL.Devices.Device(1500)
strProperties = strProperties & "DevID = "& _
CStr(DevObject.DevID)
strProperties = strProperties & "PID = "& CStr(DevObject.PID)
strProperties = strProperties & "RF Version = "& _
CStr(DevObject.RFMaxProtocolVersion) & "." & _
CStr(DevObject.RFMinProtocolVersion)
...
```

## 7.2 METHOD

### *METHOD DATAWRITE*

#### *PURPOSE*

---

Sends data to a device.

#### *DESCRIPTION*

---

This method sends data as a string to a particular device. ***This operation is allowed only when working in the DataEvent handler.***

If the **DataWrite** command is not right, the control sends an error event (**ErrorEvent**).

The possible values for the *vt* data parameter of this method are: VT\_BSTR, VT\_UI1, VT\_I2, VT\_I4, VT\_R4, VT\_R8. In all these cases the control internally converts data to be sent to an array of bytes having length *Len*.

#### *IDL*

---

```
[id(7), helpstring("method DataWrite")] HRESULT DataWrite([in] VARIANT Data, [in] short Len, [out,retval] VARIANT_BOOL *bRet);
```

#### *INPUT PARAMETERS*

---

Data as VARIANT

Len as Short

#### *OUTPUT VALUE*

---

BOOL (TRUE if the sending data thread has been successfully created)

*EXAMPLE*

```

...
...
/ Data Event Handler In this example if kind value is TRUE
/ the frame is two ways, so
/ the device is waiting for a response from the PC
/
Private Sub RFNCCTL_DataEvent(ByVal SatID As Integer, ByVal
DevID As Integer, ByVal Kind As Boolean)
On Error Resume Next
Dim item As ListItem
Dim sDevice As Device
Dim vaData As Variant
Dim strData As String
.....
    Set sDevice = OCX.Devices.Device(DevID)
    vaData = sDevice.DeviceData

    ,
    'Data interpretation
    ,
.....
    ,
    'Answer manager
    ,
    If Cbool(Kind) = True Then
        item.ListSubItems.Add , , "Two Way"
        'Send back the string "OK"
        strAnswer = "OK"
        If Not sDevice.DataWrite(strAnswer, Len(strAnswer))=True
        Then ListData.ListItems.Add , , "Error Send Data"
    Else
        item.ListSubItems.Add , , "One Way"
    End If
End Sub
...
...

```

## 8 ERROR CODES

---

### 8.1 CONTROL ERROR CODES

|      |                                                            |
|------|------------------------------------------------------------|
| 0001 | Error on closing network                                   |
| 0002 | Error on opening serial line                               |
| 0003 | Error on opening network                                   |
| 0004 | Error on initializing library functions                    |
| 0005 | Network already opened                                     |
| 0006 | Timeout out of range                                       |
| 0007 | Baud rate out of range                                     |
| 0008 | Communication port out of range                            |
| 0009 | Wrong parameters                                           |
| 0010 | Base station already exists                                |
| 0011 | Base station does not exist                                |
| 0012 | Network is still running                                   |
| 0013 | Error on setting base station state                        |
| 0014 | Error on sending data                                      |
| 0015 | Error on sending Alive command to base station             |
| 0016 | Error on getting configuration                             |
| 0017 | Error on getting base station state                        |
| 0018 | Error no base station defined                              |
| 0019 | Error on sending configuration to base station             |
| 0020 | Device already exists                                      |
| 0021 | Error on getting status from base station                  |
| 0022 | Error on setting event timer                               |
| 0023 | Error on creating RF Network Protocol Handle               |
| 0024 | Error on reading data from buffer                          |
| 0025 | Error on adding device to the collection                   |
| 0026 | Error on adding base station to the collection             |
| 0027 | Error on reading control configuration                     |
| 0028 | Error on writing control configuration                     |
| 0029 | Error base station is off                                  |
| 0030 | Error on starting polling: serial communication not active |
| 0031 | Error on starting polling: no base station defined         |
| 0032 | Error on creating polling thread                           |
| 0033 | Error on closing serial communication                      |
| 0034 | Wrong base station address                                 |
| 0035 | Wrong device address                                       |

## 8.2 NETWORK LAYER DLL ERROR CODES

|      |                                                                  |
|------|------------------------------------------------------------------|
| 1000 | Error on creating thread synchronization object                  |
| 1001 | Error on opening the serial communication                        |
| 1002 | Pipe error                                                       |
| 1003 | No base station is defined                                       |
| 1004 | One or more base stations having a wrong address                 |
| 1005 | Error on creating thread                                         |
| 1006 | Error on getting base station internal descriptor                |
| 1007 | Error on getting device internal descriptor                      |
| 1008 | Wrong base station address                                       |
| 1009 | Wrong device address                                             |
| 1010 | Unknown frame type                                               |
| 1011 | Polling error: frame length >0 and <5                            |
| 1012 | No more in polling state                                         |
| 1013 | Error: line closed                                               |
| 1014 | Error on creating command thread                                 |
| 1015 | Double Command Thread ==> Exit                                   |
| 1016 | Command Alive NOT RIGHT                                          |
| 1017 | Command Status NOT RIGHT                                         |
| 1018 | Command Send Cfg NOT RIGHT                                       |
| 1019 | Command Get Cfg NOT RIGHT                                        |
| 1020 | Command HW test NOT RIGHT                                        |
| 1021 | Unknown command                                                  |
| 1022 | Sent Data NOT RIGHT                                              |
| 1023 | Initial data Get Configuration from base station NOT RIGHT       |
| 1024 | Error on opening download file                                   |
| 1025 | Error on setting base station in download state                  |
| 1026 | Poll or Data error for HW test                                   |
| 1027 | Error during polling initial phase                               |
| 1028 | There is one base station with default address. Set new address. |
| 4000 | Base station returned error. Recovery failed.                    |
| 4001 | UNKNOWN SERVER answer. Base station is broken.                   |

### 8.3 CLIENT ERROR CODES

|             |                                                         |
|-------------|---------------------------------------------------------|
| <b>2002</b> | CLIENT error - Framing error                            |
| <b>2003</b> | CLIENT error - Wrong frame, incorrect format            |
| <b>2004</b> | CLIENT error – Checksum failed                          |
| <b>2005</b> | CLIENT error – Unexpected sequence number/control value |
| <b>2006</b> | CLIENT error – Unexpected source                        |
| <b>2007</b> | CLIENT error - HEX file format error                    |
| <b>2008</b> | CLIENT error - File write error                         |
| <b>2009</b> | CLIENT error - File read error                          |
| <b>2010</b> | CLIENT error - Input parameters error                   |
| <b>2011</b> | UNEXPECTED CLIENT error returned                        |

### 8.4 STARGATE™ RF BASE STATION ERROR CODES

|             |                            |
|-------------|----------------------------|
| <b>3050</b> | Out of sequence.           |
| <b>3051</b> | Unsupported CMD.           |
| <b>3052</b> | Wrong parameters.          |
| <b>3053</b> | Unexpected pack.           |
| <b>3054</b> | Out of memory.             |
| <b>3055</b> | Too long data.             |
| <b>3056</b> | Too short data.            |
| <b>3057</b> | Descriptor does not exist. |
| <b>3058</b> | Flash failure.             |
| <b>3059</b> | Add out of range.          |
| <b>3060</b> | No program.                |
| <b>3061</b> | Unsupported error code.    |

## 9 PROPERTY PAGES

### 9.1 GENERAL CONTROL PROPERTIES

This folder allows setting the communication parameters.

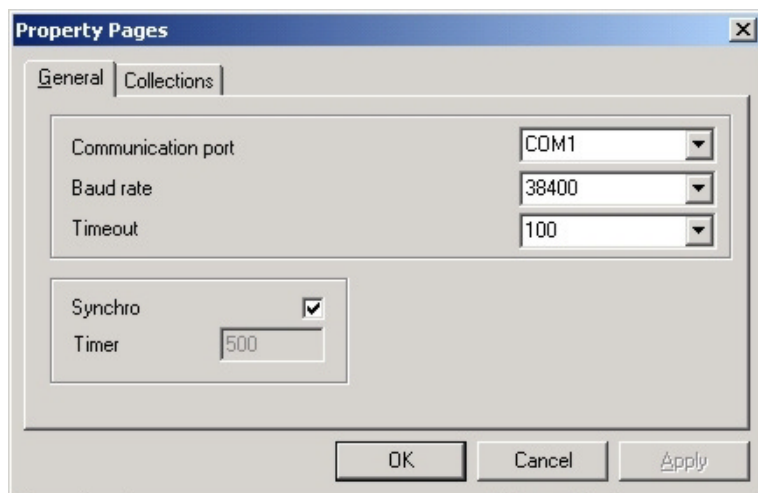


Figure 3 - General control properties

## 9.2 COLLECTION PROPERTIES

This folder allows defining the number of base stations or devices of each collection.

**NOTE**

*Adding and removing a base station or a device requires to click on the Apply button in order to update the collections.*

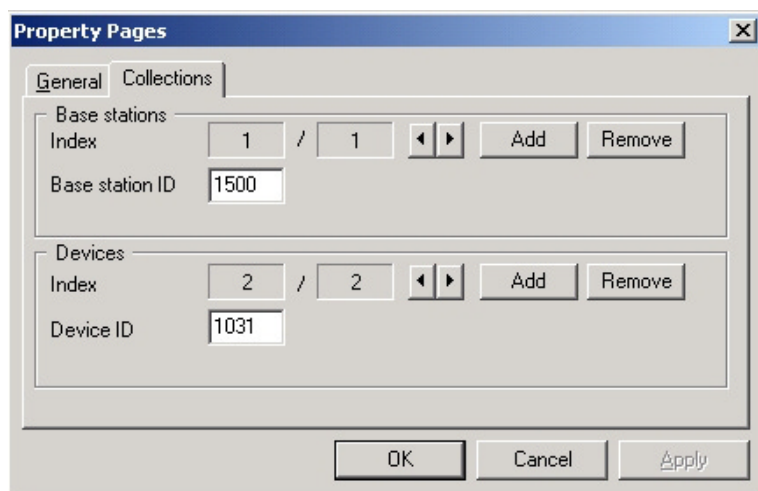


Figure 4 - Collection properties

## A PROGRAMMING EXAMPLES

### A.1 STAR-Link™ VISUAL BASIC 6.0 EXAMPLE

This example is a simple program based on STAR-Link™ ActiveX, that allows to:

- configure an RS485 base station network or a single base station in RS232
- save/load a network configuration from file (file RFNCCTL.CFG)
- start/stop PC/base station data collection
- collect data in a list
- write the main network events in a log file (file NRFCCTL.LOG)
- show data of PDF417 codes containing more than 240 characters and read by a Gryphon™ M200. The code reconstruction is performed by Visual Basic source code when Gryphon™ M200 has RF protocol version prior to 1.12. If using a Gryphon™ M200 with an RF protocol version 1.12 or later, the code reconstruction will be performed by STAR-Link™ ActiveX. To get the current RF protocol version of your RF hand-held reader, refer to the values of the RFMinProtocolVersion and RFMaxProtocolVersion properties notified during the device registration.

The main STAR-Link™ ActiveX events implemented in this example are: DataEvent, ErrorEvent, AliveEvent, SatCfgEvent, StatusEvent, SleepSatEvent, LostFrameEvent.

At start up the program opens a dialog asking whether loading an existing configuration:

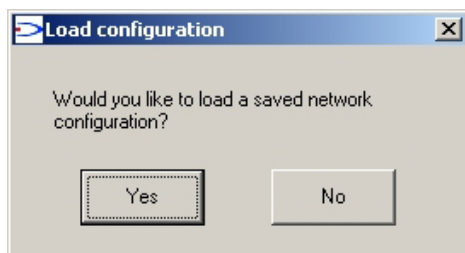


Figure 5 - Loading an existing configuration

This allows to have always a configuration reporting all information about the network devices.

For the same reason, another dialog box has been enabled when closing a configuration to save it.



### A.1.1 Network Menu

The **Network** menu provides the following options:

- **Open:** starts the PC/base station data collection displayed in the data sheet. It can be also chosen by clicking on the  icon. Before starting the collection a dialog box appears allowing to load a system configuration previously saved.
- **Close:** ends the PC/base station data collection. It can be also chosen by clicking on the  icon. Before ending the data collection a dialog box appears allowing to save the current system configuration.
- **Status:** gets the network status. It may send the following messages:
  - Serial line OPENED: at program start up when the network has not started to work;
  - Network is RUNNING: when data collection is active;
  - Network is IDLE: when data collection is off and it has been started at least one time.The network status is also shown on the status bar.
- **Exit:** after making sure that the data collection has been completed, select this command to exit the program. It can be also chosen by clicking on the  icon.

## A.1.2 Settings Menu

Before starting the data collection, configure the network by means of the menu options as follows:

- Set the address and the interface (RS-232 or RS-485) of each base station;
- Set the address of each device sending data packets to the base stations;
- Set the serial port parameters;
- Save/Load the network configuration and parameters, if desired.

The **Settings** menu provides the following options:

- **Base Stations:** by choosing this option or clicking on the  icon the following dialog box appears:

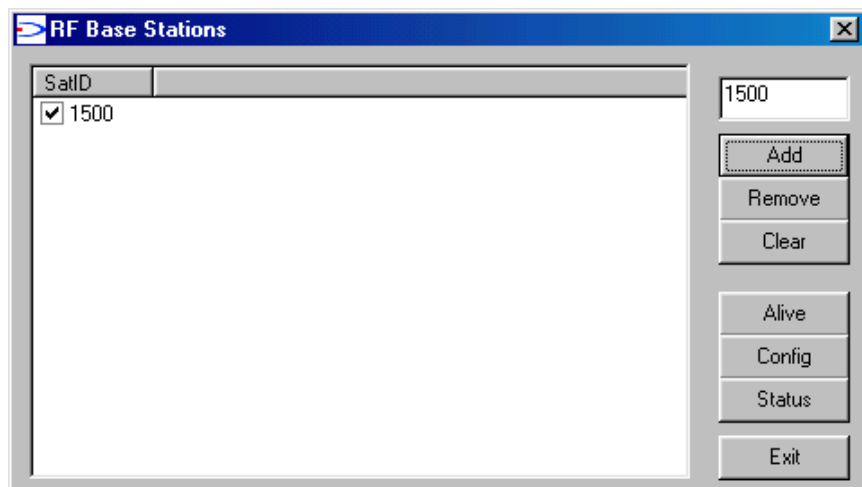


Figure 7 - Base station network

This window shows the set of base station addresses to be polled, while the check box positioned on the left side of each address allows selecting the relative base station for the network data collection. When starting the collection all the addresses listed in the window must be selected by ticking the corresponding check box. If this procedure fails, the data collection will not start and an error message appear.

Once the data collection is in progress, it is possible to suspend it on a single base station by ticking or not the corresponding check box.

**Add** button: adds a new base station in the network once its address (SatID) has been inserted in the edit box positioned on the upper right side of the dialog box (in Figure 7 the base station ID is 1500).

**Remove** button: removes one or more specific base stations or the first one from the list if no base station has been selected. It is possible to select the base station(s) to be removed by ticking the corresponding check box.

**Clear** button: removes all base stations from the list.

**Alive** button: calls the selected base station (or the first one of the list if no base station has been selected). If active, a string reporting the base station software version appears.

**Config** button: opens a dialog box showing the whole set of configuration parameters of the base station:

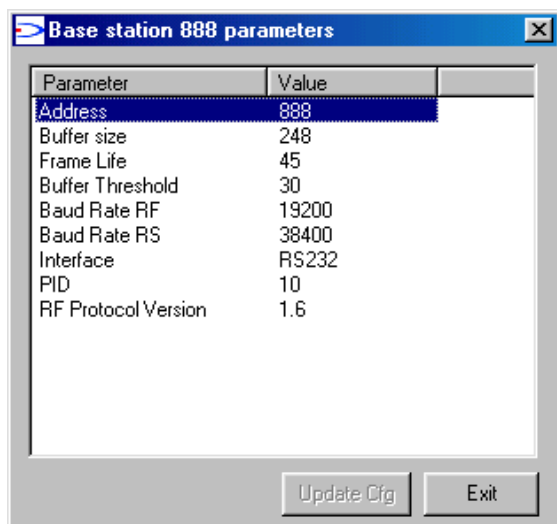


Figure 8 - Base station configuration

It is possible to modify the parameter value but it is strongly recommended to **modify** only Address and Interface (RS-232 or RS-485). To confirm the new values click on the **UpdateCfg** button through which the new configuration is sent to the base station.

**Status** button: shows the state of the receiving and transmission buffer of the selected base station (or the first one of the list if no base station has been selected).

**Exit** button: closes the dialog and saves the network configuration for the current session. It is possible to save the configuration to file by selecting the Save Configuration option from the Settings menu.

- **Devices:** by choosing this option or clicking on the  icon the following dialog box appears:

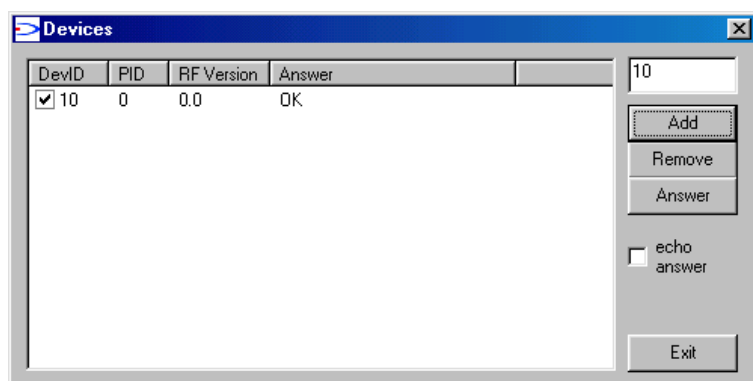


Figure 9 - Device network

The check box positioned on the left side of each address allows sending or not data packets from that device to the application. If the check box corresponding to the desired device is not ticked, the application will not receive data anymore.

**Add** button: has the same function as described for the base station management.

**Remove** button: has the same function as described for the base station management.

**Answer** button: allows setting a string to answer to the devices transmitting in 2 way mode.

**Check “echo Answer”:** by ticking the check box the program sends the same frame previously received by a device working in 2 way mode. If the check box is not ticked, the program will send the OK default answer (as shown in Figure 9).

**Exit** button: saves changes for the current session and closes the dialog box. It is possible to save the configuration to file by selecting the Save Configuration option from the Settings menu.

**PID** and **RF Version** provide information the program gets from the first frame sent by the device to the network.

- **Load configuration:** gets the network configuration from the RFNCCTL.CFG

file. It can be also chosen by clicking on the  icon. The corresponding dialog box automatically appears when starting the data collection (see par. A.1).

- **Save configuration:** saves the network configuration to the RFNCCTL.CFG file.

It can be also chosen by clicking on the  icon. The corresponding dialog box is automatically enabled when closing a configuration (see par. A.1):

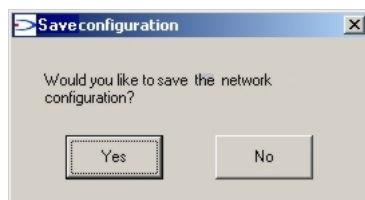


Figure 10 - Saving a network configuration

- **Control settings:** allows setting the serial communication parameters, a synchronous or asynchronous behaviour (and timer if asynchronous) in answering data to devices in 2 way transmission mode (see property Synchro). Then, it allows defining the mode to interpret data (see property Mode) and setting the program behaviour in case of frame loss (see SetLostFrameBehaviour method).

The control settings must be defined only when the network data collection is stopped or completed.

This option can be also chosen by clicking on the  icon.

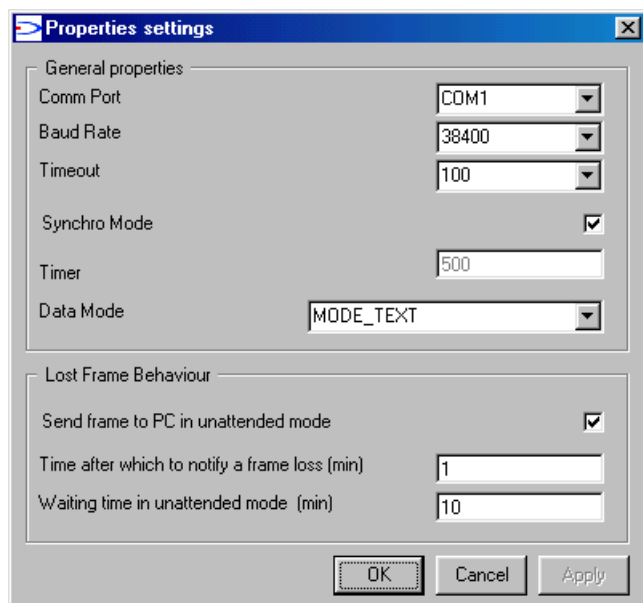


Figure 11 - Control settings

**Apply** button: saves changes before closing the dialog.

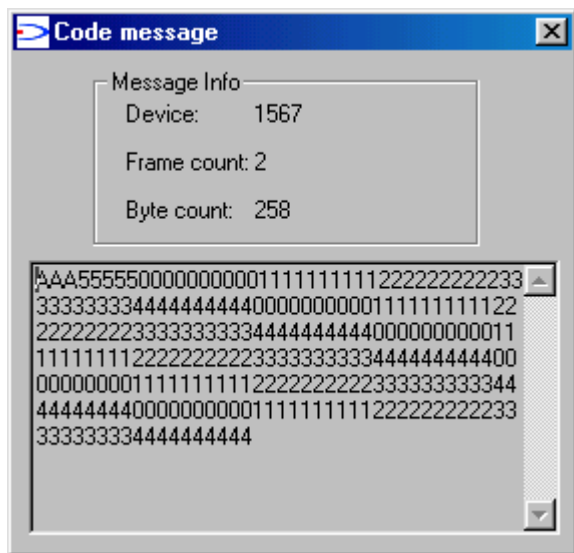
**OK** button: saves and closes the dialog.

- **Configuration path:** sets the RFNCCTL.CFG path. Load and Save commands search for the RFNCCTL.CFG in the indicated path.

### A.1.3 Action Menu

The **Action** menu provides the following options:

- **Show log:** opens the NRFCCTL.LOG file reporting the most important network actions.
- **Background data collection:** reduces the running program to an icon on the Windows toolbar.
- **Always on top:** causes the program window to be always in foreground.
- **Show long code (PDF417):** opens a dialog box showing data of codes read by a Gryphon™ M200 including those containing more than 240 characters (ex. PDF417):



### Figure 12 - Code message

**Frame Count:** shows the number of data packets containing the code message. Each data packet consists of a maximum of 240 characters.

**Byte Count:** shows the total number of character contained in the code message.

### Example of message reconstruction from Gryphon™ M200

This is a simple example providing the RF device and radio protocol configuration to reconstruct a message containing more than 240 characters and split into different data packets.

Since the data packets transmitted during each transaction consist of a maximum of 240 characters, it is necessary to split the original message into several data packets when reading a code having more than 240 characters.

If reading a PDF417 code with Gryphon™ M200 it is possible to split the code message by following the given procedure:

1. Configure the Gryphon™ M200 by following the steps described in the "Update for Gryphon™ M" document and setting the "Set Connection" (step 3) to value 2;
2. Set the STAR-Link™ ActiveX to **Binary Mode**.

#### A.1.4 Help Menu

The ? (Help) menu provides the following options:

- **About VB Example:** opens a dialog box showing information about the program and its version number.
- **About Network manager DLL:** shows the DLL version number.
- **About STAR-Link™ ActiveX:** shows the STAR-Link™ version number.

### A.1.5 Warning Messages

In the following cases a warning dialog box to be set by the user appears:

1. if some data packets have been lost, the program opens a dialog box asking if the data collected in the base station after the loss should be sent to the PC or be deleted. For more details see par. 3.2 and par. 3.3, in particular `tLostFrameBehaviour`, `GetLostFrameBehaviour`, `LostFrame` method and `LostFrameEvent` event.

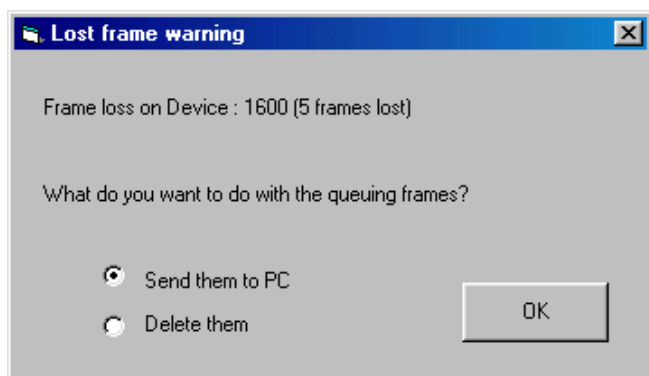


Figure 13 - Lost frame warning

2. if one base station does not answer to the PC, the program opens a dialog box asking if it is necessary to set the base station state off since it is damaged (in this way the satellite is no longer polled), or not. If the state does not change, this dialog box may appear again after few attempts to poll the base station. For more details see par. 3.3 , in particular the `SleepSatEvent` event.



Figure 14 - Sleeping base station warning

## A.2 STAR-Link™ EXCEL 97 EXAMPLE

This is a simple example of configuration and data collection based on STAR-Link™ ActiveX.

A VBA macro uses OpenNetwork, CloseNetwork, Network Status methods, Device DataWrite method, DataEvent and ErrorEvent exported by STAR-Link™ ActiveX (see pars. 3.1, 3.2, 3.3 and 7.2 ).

To open the Excel sheet implementing the example click on the "Enable Macros" button of the opening dialog box. Then, the following window appears:

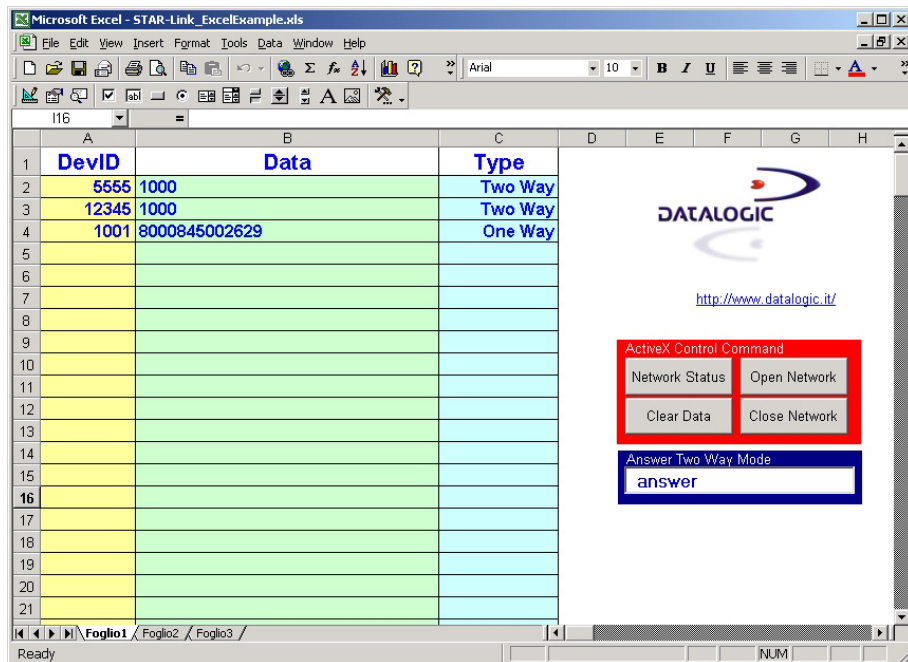


Figure 15 - Main window

On the left side there is a database consisting of three columns reporting data information from frames (by collecting more than one thousand packets the list will be reset):

- **DevID** column: lists the device identification number;
- **Data** column: lists the data packet
- **Type** column: shows if transaction is 1way or 2 ways.

On the right side there are four buttons and an edit box:

- **Network Status** button: shows if the network data collection is in progress or not (idle);
- **Clear Data** button: removes all data frames collected in the **Data** column;
- **Open Network** button: starts the base station data collection;
- **Close Network** button: stops the base station data collection.
- **Answer two way mode** edit box: contains the string to be sent back to each device waiting for an answer from the PC.

## A.2.1 Network Configuration

The network base stations and devices can be set in the Visual Basic application window. To open this window it is necessary to click at first on the design mode icon; then, to double-click on the STAR-Link™ ActiveX button. The complete toolbar can be displayed by following the path: Tools→Customize→Toolbars→Visual Basic.

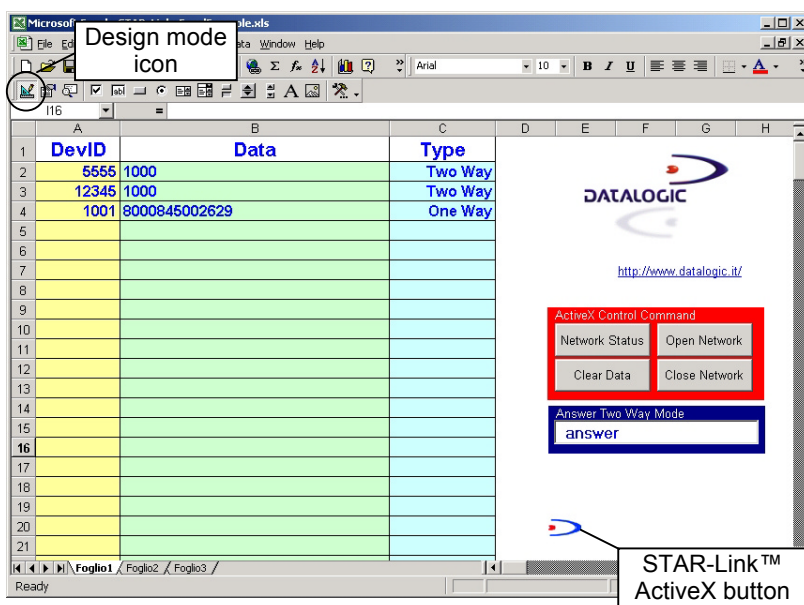


Figure 16 - Configuration window

Once the Visual Basic Application project has been opened, the following window appears:

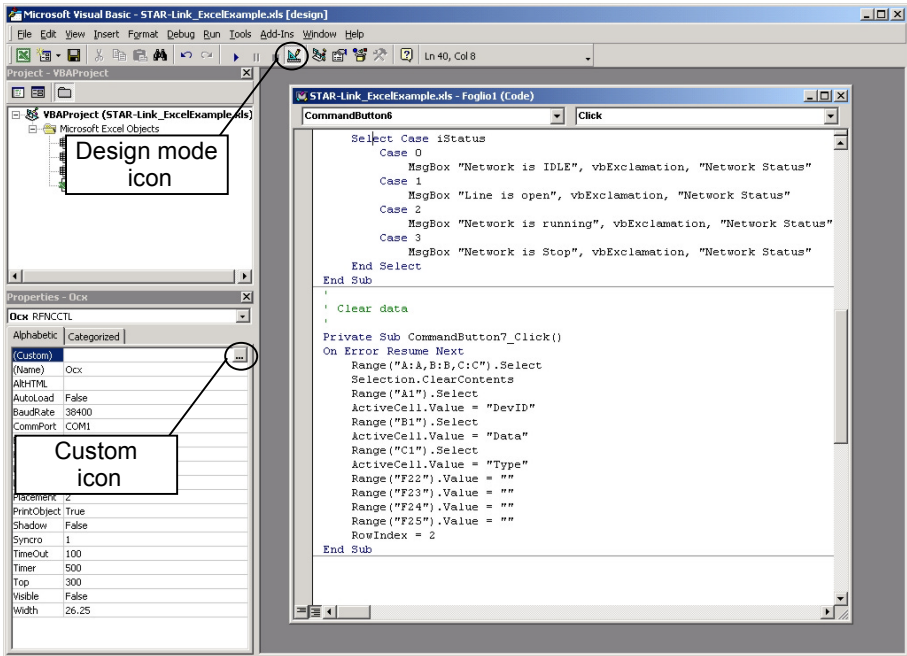


Figure 17 - Application project

By double-clicking on the Custom item of the ActiveX object in the project, the following dialog box appears:

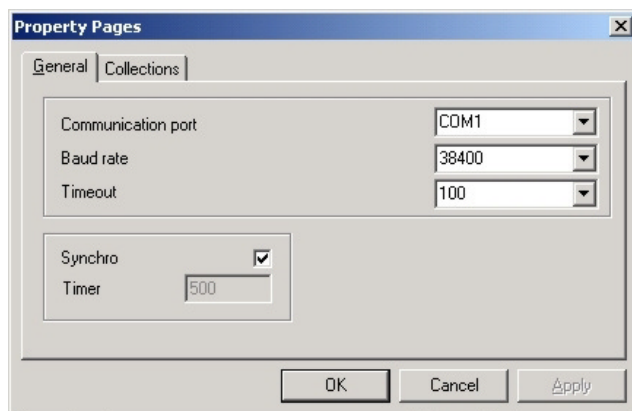


**Figure 18 - Property window**

These property pages (base stations and device collections, general properties) allow setting the network configuration, which can be saved by proceeding as follow:

- Click on the **Apply** button;
- Close the VBA;
- Return to the Excel data sheet;
- Save the configuration by selecting the **Save** option from the **File** menu;
- Activate the macro again.

The advised procedure is to close, save, return to Excel and open again: the program will asks for the macro activation again.

### A.3 STAR-Link™ ACCESS 97 EXAMPLE

The example is a simple Access program based on STAR-Link™ ActiveX. It allows configuring the base station and device network and collecting data and quantity. A product database is also provided so that, if a barcode identifying a product present in database is scanned and the device is transmitting data in 2 way mode, the program selects a particular data present in the selected field and sends it back to the device:

**CAUTION**

*To run this program in a Windows 2000 environment you must choose the "Convert Database" option from the opening dialog box. Then, it is also possible to change the filename.*

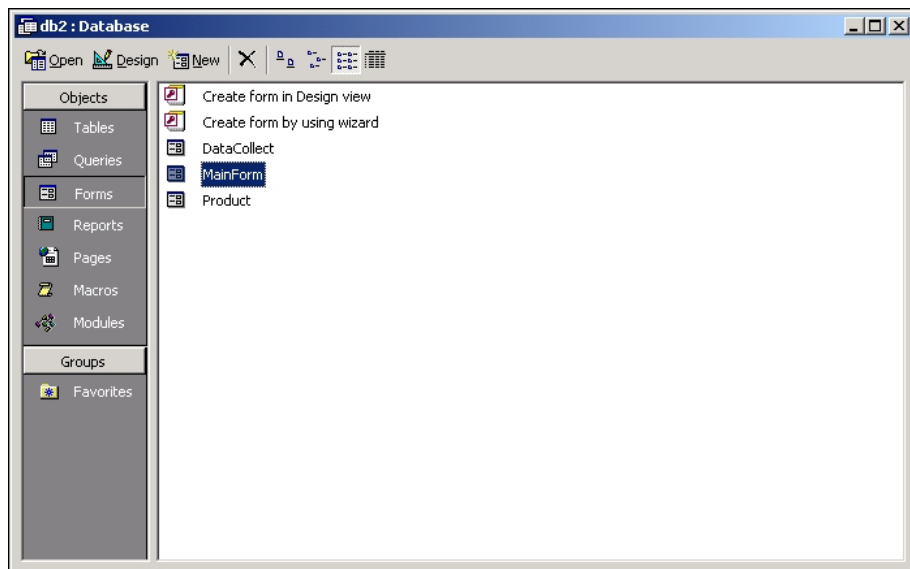


Figure 19 - Main window

### A.3.1 MainForm Mask

By choosing **MainForm** from the mask folder and clicking on the **Open** button, the program starts and the following window appears:

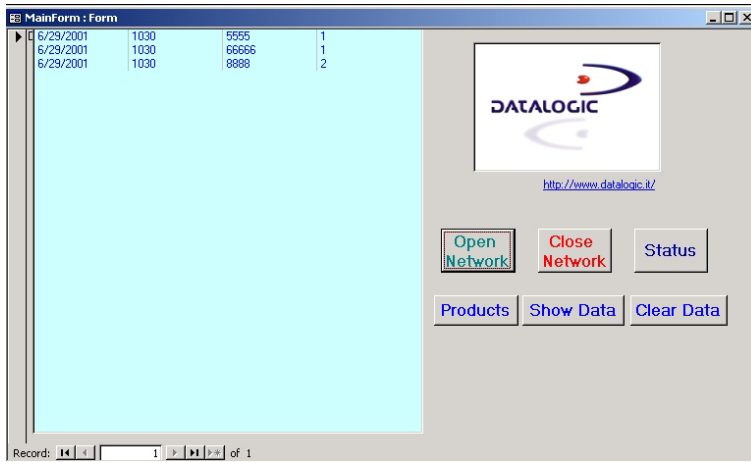


Figure 20 - MainForm mask

On the right side there are the following buttons:

- **Open Network** button: starts network data collection;
- **Close network** button: stops network data collection;
- **Status** button: shows the network state;
- **Products** button: shows product database;
- **Show Data** button: shows all data of the table;
- **Clear Data** button: clears all data collected in the Data Collect table on the left side.

On the left side all the collected data are ordered in the Date, Device Identification Number, Data and Quantity field (by collecting more than one thousand packets the list will be reset). The green or red color of this area indicates respectively that the network is open or closed.

Before starting the data collection, the network must be configured. As already described in the Excel example, double-click on the control ActiveX button visible in design mode (after clicking on **Design**). Then, the property page dialog box appears allowing to configure the network through the **General** and **Collections** folder (see Chapter 9).

## A.3.2 Other Masks

By choosing **DataCollect** or **Product** from the mask folder and clicking on the **Open** button, you get the following information:

- **Data Collected:** it is a database consisting of data packets and device information sent by the devices. The progressive ID, DevID, Data, Quantity, Date fields are displayed on the left side of the MainFrom window. The Quantity field is incremented every time a device sends the same data packet.
- **Product:** the database fields are ID, Data, Price, Description. The Description field content is, in this example, the data returned in a two way transaction to the device.

As in the Excel example, the VBA source code uses OpenNetwork, CloseNetwork, Network Status methods, Device DataWrite method, DataEvent and ErrorEvent exported by ActiveX (see pars. 3.1, 3.2, 3.3 and 7.2).

## A.4 STAR-Link™ C++ BUILDER 5 EXAMPLE

The example is a simple program based on STAR-Link™ ActiveX, that allows to:

- configure an RS485 base station network or a single base station in RS232
- save/load a network configuration from file (file RFNCCTL.CFG)
- start/stop PC/Base station data collection
- collect data in a list
- write the main network events in a log file (file NRFCCTL.LOG)

The main STAR-Link™ ActiveX events implemented in this example are: DataEvent, ErrorEvent, AliveEvent, SatCfgEvent, StatusEvent, SleepSatEvent, LostFrameEvent.

At start up the program opens a dialog providing a data sheet to collect the data packets received from the devices:

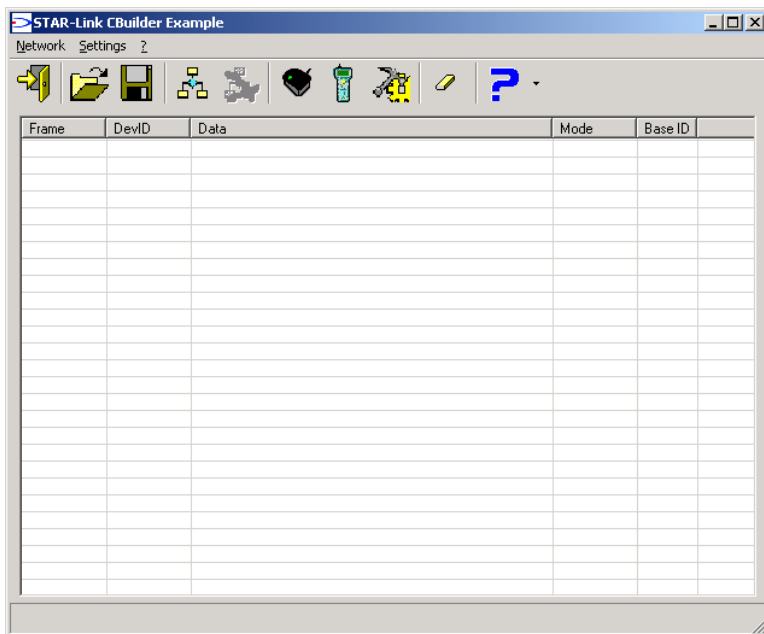


**Figure 21 - Main window**

Refer to the Visual Basic example for button description, since they share the same functions apart from the answer string.

## A.5 STAR-Link™ DELPHI 6 EXAMPLE

The example is a simple program based on STAR-Link™ ActiveX, that allows to:

- add base stations and devices to the network
- save/load a network configuration from file (file RFNCCTL.CFG)
- start/stop PC/Base station data collection
- collect data in a list
- write the main network events in a log file (file NRFCCTL.LOG)

The main STAR-Link™ ActiveX events implemented in this example are: DataEvent, ErrorEvent, AliveEvent.

At start up the program opens the following dialog providing a data sheet to collect the data packets received from the device:

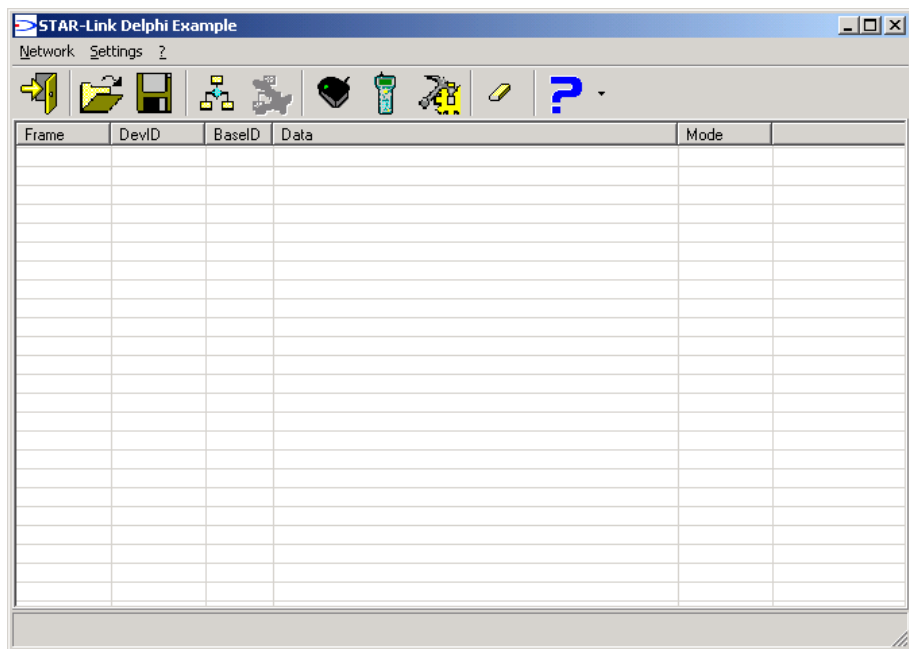


**Figure 22 - Main window**

Refer to the Visual Basic example for button description, since they share the same functions.

## B CONTROL INTERFACE

This .idl file provides an integral interface exposed from the control in IDL notation.

```

/*****
* Name ..... NRFCCtlControl.idl
* .....
* Author ..... bFM
* Date ..... 12/04/02
* Description ..... IDL source for NRFCCtlControl.dll
*
*           This file will be processed by the MIDL tool to
*           produce the type library (NRFCCtlControl.tlb)
*           and marshalling code.
*
*****/

import "oidl.idl";
import "ocidl.idl";
#include "olectl.h"

interface ISats;
interface ISat;
interface IDevices;
interface IDevice;

[
    object,
    uuid(CF80214F-0D60-11D5-A274-005004C31B96),
    dual,
    helpstring("IRFNCCTL Interface"),
    pointer_default(unique)
]
interface IRFNCCTL : IDispatch
{
    [propget, id(DISPID_ENABLED)] HRESULT Enabled([out, retval] VARIANT_BOOL
*pbool);
    [propput, id(DISPID_ENABLED)] HRESULT Enabled([in] VARIANT_BOOL vbool);
    [propget, id(1), helpstring("property BaudRate")] HRESULT BaudRate([out, retval]
BSTR *pVal);
    [propput, id(1), helpstring("property BaudRate")] HRESULT BaudRate([in] BSTR
newVal);
    [propget, id(2), helpstring("property Satellites")] HRESULT Satellites([out, retval] ISats*
*pVal);
    [propget, id(3), helpstring("property Devices")] HRESULT Devices([out, retval]
IDevices* *pVal);
    [id(4), helpstring("method OpenNetwork")] HRESULT OpenNetwork([out, retval]
VARIANT_BOOL *retVal);
    [id(5), helpstring("method CloseNetwork")] HRESULT CloseNetwork([out, retval]
VARIANT_BOOL *retVal);
    [propget, id(6), helpstring("property CommPort")] HRESULT CommPort([out, retval]
BSTR *pVal);
    [propput, id(6), helpstring("property CommPort")] HRESULT CommPort([in] BSTR
newVal);

```

```

        [propget, id(7), helpstring("property TimeOut")] HRESULT TimeOut([out, retval] short
*pVal);
        [propput, id(7), helpstring("property TimeOut")] HRESULT TimeOut([in] short newVal);
        [id(8), helpstring("method OpenLine")] HRESULT OpenLine([out, retval]
VARIANT_BOOL *retVal);
        [id(9), helpstring("method CloseLine")] HRESULT CloseLine([out, retval]
VARIANT_BOOL *retVal);
        [id(11), helpstring("method Load")] HRESULT Load([optional]VARIANT Path,[out,
retval] VARIANT_BOOL *retVal);
        [id(12), helpstring("method Save")] HRESULT Save([optional]VARIANT Path,[out,
retval] VARIANT_BOOL *retVal);
        [propget, id(13), helpstring("property NetworkStatus")] HRESULT NetworkStatus([out,
retval] short *pVal);
        [id(14), helpstring("method GetDLLVersion")] HRESULT
GetDLLVersion([out,retval]BSTR *Version);
        [propget, id(15), helpstring("property Synchro")] HRESULT Synchro([out, retval] BOOL
*pVal);
        [propput, id(15), helpstring("property Synchro")] HRESULT Synchro([in] BOOL
newVal);
        [propget, id(16), helpstring("property Timer")] HRESULT Timer([out, retval] short
*pVal);
        [propput, id(16), helpstring("property Timer")] HRESULT Timer([in] short newVal);
        [id(17), helpstring("method GetControlVersion")] HRESULT
GetControlVersion([out,retval]BSTR *Version);
        [id(18), helpstring("method SetLostFrameBehaviour")] HRESULT
SetLostFrameBehaviour(VARIANT_BOOL Send2PC, short TimeNotify, short TimeGoingOn);
        [id(19), helpstring("method GetLostFrameBehaviour")] HRESULT
GetLostFrameBehaviour([out]VARIANT_BOOL *Send2PC, [out]short* TimeNotify, [out]short* TimeGoingOn);
        [id(20), helpstring("method LostFrame")] HRESULT LostFrame([in]VARIANT_BOOL
Send2PC, [out,retval]VARIANT_BOOL* retVal);
        [propget, id(21), helpstring("property Mode")] HRESULT Mode([out, retval] short
*pVal);
        [propput, id(21), helpstring("property Mode")] HRESULT Mode([in] short newVal);
    };

[
    uuid(CF802143-0D60-11D5-A274-005004C31B96),
    version(1.4),
    helpstring("STAR-Link ATL Control 1.4")
]
library NRFCCTLCONTROLLib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    [
        uuid(CF802151-0D60-11D5-A274-005004C31B96),
        helpstring("_IRFNCCTLEvents Interface")

        dispinterface _IRFNCCTLEvents
        {
            properties:
            methods:
            [id(1), helpstring("method ErrorEvent")] HRESULT ErrorEvent(short ErrorID, BSTR
ErrorDesc);
            [id(2), helpstring("method DataEvent")] HRESULT DataEvent(short SatID, short DevID,
BOOL Kind);
            [id(3), helpstring("method AliveEvent")] HRESULT AliveEvent(short SatID, BSTR Info);
            [id(4), helpstring("method SatCfgEvent")] HRESULT SatCfgEvent(short SatID);
            [id(5), helpstring("method StatusEvent")] HRESULT StatusEvent(short SatID);

```

```

short Status);
[id(6), helpstring("method SleepSatEvent")] HRESULT SleepSatEvent(short SatID,
long NumLostFrame);
[id(7), helpstring("method LostFrameEvent")] HRESULT LostFrameEvent(short DevID,
);
[id(8), helpstring("method DeviceInfoEvent")] HRESULT DeviceInfoEvent(short DevID);
{
    object,
    uuid(CF802152-0D60-11D5-A274-005004C31B96),
    dual,
    helpstring("ISats Interface"),
    pointer_default(unique)
}
interface ISats : IDispatch
{
    [propget, id(DISPID_NEWENUM)] HRESULT _NewEnum([out, retval] LPUNKNOWN
*pUnk);
    [propget, id(DISPID_VALUE)] HRESULT Item([in]long Index, [out, retval] VARIANT
*pVal);
    [id(1), helpstring("method Add")] HRESULT Add([in]VARIANT SatID, [out, retval] ISat
**pVal);
    [id(2), helpstring("method Remove")] HRESULT Remove([in]VARIANT SatID;
    [propget, id(3), helpstring("property Count")] HRESULT Count([out, retval] long *pVal);
    [propget, id(4)] HRESULT Satellite([in]VARIANT SatID, [out, retval] ISat **pVal);
    [id(5), helpstring("method Clear")] HRESULT Clear();
};
{
    object,
    uuid(CF802154-0D60-11D5-A274-005004C31B96),
    dual,
    helpstring("ISat Interface"),
    pointer_default(unique)
}
interface ISat : IDispatch
{
    [propget, id(1), helpstring("property SatID")] HRESULT SatID([out, retval] short *pVal);
    [propput, id(1), helpstring("property SatID")] HRESULT SatID([in] short newVal);
    [propget, id(2), helpstring("property Enabled")] HRESULT Enabled([out, retval] BOOL
*pVal);
    [propput, id(2), helpstring("property Enabled")] HRESULT Enabled([in] BOOL newVal);
    [propget, id(3), helpstring("property PID")] HRESULT PID([out, retval] short *pVal);
    [propput, id(3), helpstring("property PID")] HRESULT PID([in] short newVal);
    [propget, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT
RFMaxProtocolVersion([out, retval] short *pVal);
    [propput, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT
RFMaxProtocolVersion([in] short newVal);
    [propget, id(5), helpstring("property RFMinProtocolVersion")] HRESULT
RFMinProtocolVersion([out, retval] short *pVal);
    [propput, id(5), helpstring("property RFMinProtocolVersion")] HRESULT
RFMinProtocolVersion([in] short newVal);
    [propget, id(6), helpstring("property BaudRateRS")] HRESULT BaudRateRS([out,
retval] long *pVal);
    [propput, id(6), helpstring("property BaudRateRS")] HRESULT BaudRateRS([in] long
newVal);
    [propget, id(7), helpstring("property BaudRateRF")] HRESULT BaudRateRF([out,
retval] long *pVal);
    [propput, id(7), helpstring("property BaudRateRF")] HRESULT BaudRateRF([in] long
newVal);
    [propget, id(8), helpstring("property BufferSize")] HRESULT BufferSize([out, retval]
short *pVal);

```

```

newVal);
[propput, id(8), helpstring("property BufferSize")] HRESULT BufferSize([in] short
retval] short *pVal);
[propget, id(9), helpstring("property BufferThreshold")] HRESULT BufferThreshold([out,
[propput, id(9), helpstring("property BufferThreshold")] HRESULT BufferThreshold([in]
short newVal);
[propget, id(10), helpstring("property Direct")] HRESULT Direct([out, retval] BOOL
*pVal);
[propput, id(10), helpstring("property Direct")] HRESULT Direct([in] BOOL newVal);
[propget, id(11), helpstring("property FrameLife")] HRESULT FrameLife([out, retval]
short *pVal);
[propput, id(11), helpstring("property FrameLife")] HRESULT FrameLife([in] short
newVal);
[propget, id(12), helpstring("property Interface")] HRESULT Interface([out, retval] short
*pVal);
[propput, id(12), helpstring("property Interface")] HRESULT Interface([in] short
newVal);
[propget, id(13), helpstring("property MaxBackOff")] HRESULT MaxBackOff([out,
retval] short *pVal);
[propput, id(13), helpstring("property MaxBackOff")] HRESULT MaxBackOff([in] short
newVal);
[propget, id(14), helpstring("property MinBackOff")] HRESULT MinBackOff([out, retval]
short *pVal);
[propput, id(14), helpstring("property MinBackOff")] HRESULT MinBackOff([in] short
newVal);
[propget, id(15), helpstring("property PowerSave")] HRESULT PowerSave([out, retval]
BOOL *pVal);
[propput, id(15), helpstring("property PowerSave")] HRESULT PowerSave([in] BOOL
newVal);
[propget, id(16), helpstring("property PreambleLength")] HRESULT
PreambleLength([out, retval] short *pVal);
[propput, id(16), helpstring("property PreambleLength")] HRESULT
PreambleLength([in] short newVal);
[propget, id(17), helpstring("property RxFree")] HRESULT RxFree([out, retval] short
*pVal);
[propput, id(17), helpstring("property RxFree")] HRESULT RxFree([in] short newVal);
[propget, id(18), helpstring("property RxTotal")] HRESULT RxTotal([out, retval] short
*pVal);
[propput, id(18), helpstring("property RxTotal")] HRESULT RxTotal([in] short newVal);
[propget, id(19), helpstring("property TxFree")] HRESULT TxFree([out, retval] short
*pVal);
[propput, id(19), helpstring("property TxFree")] HRESULT TxFree([in] short newVal);
[propget, id(20), helpstring("property TxTotal")] HRESULT TxTotal([out, retval] short
*pVal);
[propput, id(20), helpstring("property TxTotal")] HRESULT TxTotal([in] short newVal);
[id(21), helpstring("method Alive")] HRESULT Alive();
[id(22), helpstring("method GetConfiguration")] HRESULT GetConfiguration();
[id(23), helpstring("method SetConfiguration")] HRESULT SetConfiguration();
[propget, id(24), helpstring("property NewSatID")] HRESULT NewSatID([out, retval]
short *pVal);
[propput, id(24), helpstring("property NewSatID")] HRESULT NewSatID([in] short
newVal);
[id(25), helpstring("method GetStatus")] HRESULT GetStatus();
};
[
    object,
    uuid(5950285B-0E22-11D5-A274-005004C31B96),
    dual,
    helpstring("IDevices Interface"),
    pointer_default(unique)

```

```

    }
    interface IDevices : IDispatch
    {
        [propget, id(DISPID_NEWENUM)] HRESULT _NewEnum([out, retval] LPUNKNOWN
*pUnk);
        [propget, id(DISPID_VALUE)] HRESULT Item([in]long Index, [out, retval] VARIANT
*pVal);
        [id(1), helpstring("method Add")] HRESULT Add(VARIANT DevID, [out, retval] IDevice
**pVal);
        [id(2), helpstring("method Remove")] HRESULT Remove(VARIANT DevID);
        [propget, id(3), helpstring("property Count")] HRESULT Count([out, retval] long *pVal);
        [propget, id(4)] HRESULT Device([in]VARIANT DevID, [out, retval] IDevice **pVal);
        [id(5), helpstring("method Clear")] HRESULT Clear();
    };
    [
        object,
        uuid(5950285D-0E22-11D5-A274-005004C31B96),
        dual,
        helpstring("IDevice Interface"),
        pointer_default(unique)
    ]
    interface IDevice : IDispatch
    {
        [propget, id(1), helpstring("property DevID")] HRESULT DevID([out, retval] short
*pVal);
        [propput, id(1), helpstring("property DevID")] HRESULT DevID([in] short newVal);
        [propget, id(2), helpstring("property Enabled")] HRESULT Enabled([out, retval] BOOL
*pVal);
        [propput, id(2), helpstring("property Enabled")] HRESULT Enabled([in] BOOL newVal);
        [propget, id(3), helpstring("property BufferSize")] HRESULT BufferSize([out, retval]
short *pVal);
        [propput, id(3), helpstring("property BufferSize")] HRESULT BufferSize([in] short
newVal);
        [propget, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT
RFMaxProtocolVersion([out, retval] short *pVal);
        [propput, id(4), helpstring("property RFMaxProtocolVersion")] HRESULT
RFMaxProtocolVersion([in] short newVal);
        [propget, id(5), helpstring("property RFMinProtocolVersion")] HRESULT
RFMinProtocolVersion([out, retval] short *pVal);
        [propput, id(5), helpstring("property RFMinProtocolVersion")] HRESULT
RFMinProtocolVersion([in] short newVal);
        [propget, id(6), helpstring("property PID")] HRESULT PID([out, retval] short *pVal);
        [propput, id(6), helpstring("property PID")] HRESULT PID([in] short newVal);
        [id(7), helpstring("method DataWrite")] HRESULT DataWrite([in] VARIANT Data, [in]
short Len, [out,retval] VARIANT_BOOL *bRet);
        [propget, id(8), helpstring("property DeviceData")] HRESULT DeviceData([out, retval]
VARIANT *pVal);
        [propget, id(9), helpstring("property DataLen")] HRESULT DataLen([out, retval] short
*pVal);
    };
    [
        uuid(CF802150-0D60-11D5-A274-005004C31B96),
        helpstring("STAR-Link ATL Control 1.4 Class")
    ]
    coclass RFNCCTL
    {
        [default] interface IRFNCCTL;
        [default, source] dispinterface _IRFNCCTLEvents;
    }

```

```

};
[
    uuid(CF802153-0D60-11D5-A274-005004C31B96),
    helpstring("Sats Class")
]
coclass Sats
{
    [default] interface ISats;
};
[
    uuid(CF802155-0D60-11D5-A274-005004C31B96),
    helpstring("Sat Class")
]
coclass Sat
{
    [default] interface ISat;
};
[
    uuid(5950285C-0E22-11D5-A274-005004C31B96),
    helpstring("Devices Class")
]
coclass Devices
{
    [default] interface IDevices;
};
[
    uuid(5950285E-0E22-11D5-A274-005004C31B96),
    helpstring("Device Class")
]
coclass Device
{
    [default] interface IDevice;
};
[
    uuid(AF00E481-1478-11D5-A274-005004C31B96),
    helpstring("GenProp Class")
]
coclass GenProp
{
    interface IUnknown;
};
[
    uuid(49D8EAC1-1884-11D5-A274-005004C31B96),
    helpstring("CollProp Class")
]
coclass CollProp
{
    interface IUnknown;
};
};

```